日本電気 ハンドヘルド・コンピュータ

PC-8201

N82-BASIC

活用の基礎

日本情報研究センター 斉藤 紀美子・著



日本電気
PC-8201
N82-BASIC
ハンドヘルド・コンピュータ 活用の基礎
日本情報研究センター 斉藤 紀美子・著
学研

日本電気 ハンドヘルド・コンピュータ

PC-8201
活用の基礎

日本情報研究センター　斉藤 紀美子・著

学研

日本電気 ハンドヘルド・コンピュータ

PC-8201 活用の基礎

日本情報研究センター　斉藤　紀美子・著



学研

はじめに



NEC・PC－8201は，本格的なビジネス向きハンドヘルド・コンピュータです。つまり，手で持ち運びできるポータブルなパソコンです。

パーソナルコンピュータは，名前のとおり，個人用コンピュータとして誕生し，わずか5～6年の間にOA（オフィス・オートメーション）の中核として，その活用は拡大の一途にありますが，その一方で，機器は大型化し，使い方も複雑化しつつあります。利用が高度化した分だけ価格も上昇し，専門知識も要求され，企業に導入される場合，少しもパーソナルではなくなってきているのです。

もちろん，ポケットコンピュータと呼ばれる小型のパソコンもたくさん販売されていますが，ポケットコンピュータは，小型化した分だけ性能も小さくしてしまったため，パソコン入門機，BASICプログラム独習機としての役割に終わってしまっています。BASICプログラムを学ぶのは，プログラムが趣味のマニアは別にすれば，パソコンを活用したいと考えるからでしょう。とすると，BASICをマスターした後，会社で使っているパソコンと同じような仕事をさせることができなければ，もったいないとしかいえません。

PC－8201は，ショルダーバッグに入れて持ち運べる程度の大きさであり，電車の中で膝に乗せて仕事をすることもできるものでありながら，同時に実務に活用するのに十分な機能を持っています。まだPC－8201を知らない人のために，その魅力を知ってもらいたい，また，PC－8201の活用方法に悩んでいる人のためには，操作の基礎から本格的なビジネス利用までを紹介して，使いこなしていただきたい，そんな目的で本書を書きました。PC－8201が身近にない人のために，できる限り図や実例を入れ，実際にパソコンを動かしている感じを理解してもらえるようにしました。

もちろん，PC－8201が使用できる人は，本書に沿って動かしてもらえれば，PC－8201の機能の理解と，BASICプログラミングの両方がマスターできるはずです。

日本電気 PC-8201 ハンドヘルド・コンピュータ 活用の基礎＊目次

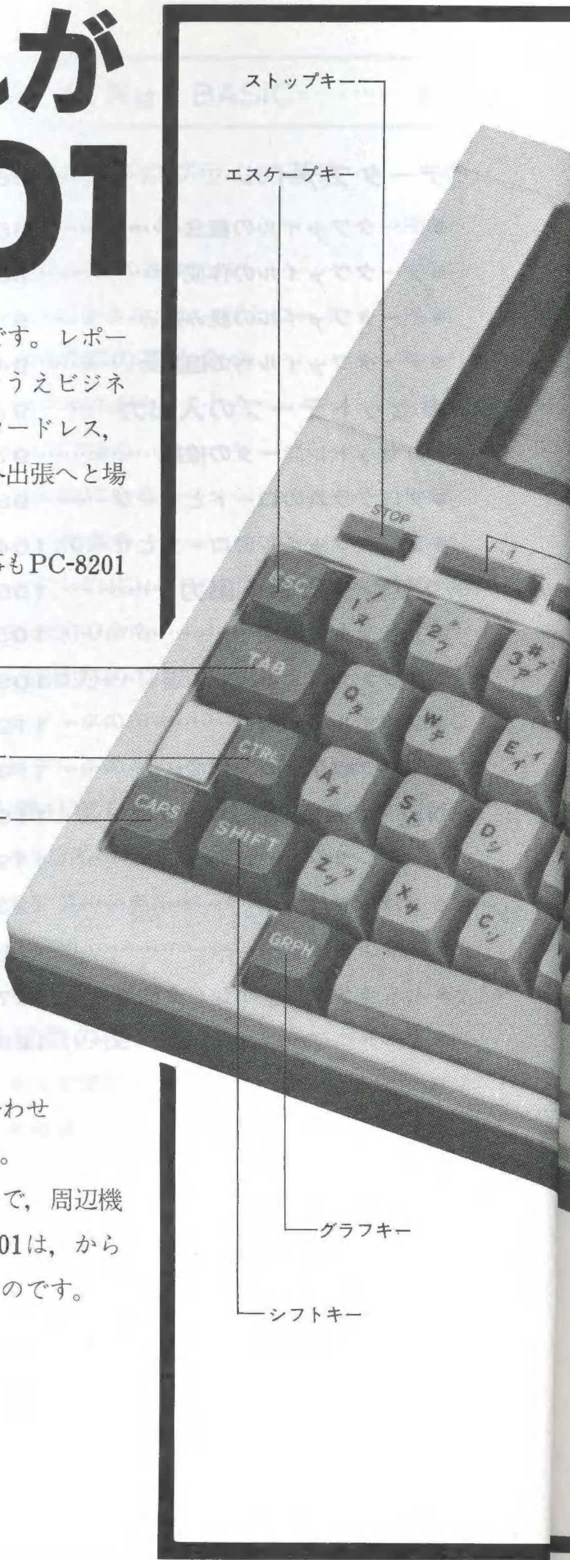
これが PC-8201

PC-8201は高性能のハンドヘルド・パソコンです。レポート用紙と同サイズのA４判。持ち運びに便利なうえビジネスに十分対応する機能がバッチリ。単３電池でコードレス，オフィスから外出先へ，フィールドワークや海外出張へと場所を選びません。

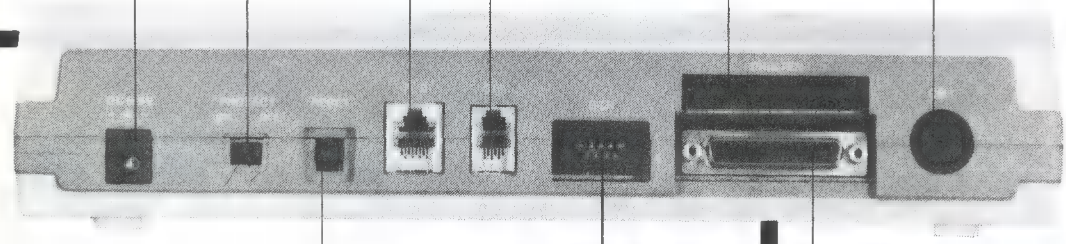
メモをしたり，原稿を書いたり……，こんな仕事もPC-8201なら簡単です。ワープロ（ワードプロセッサー）プログラム「TEXT」がROMに書きこまれているのでパソコンを手帳がわりに使えます。そしてあとからプリントアウト。文書や集計表の作成も大幅能率アップ。プログラムは一度記憶させれば最大63本までスイッチを切っても26日間は消えません。必要なときに，必要なプログラムを，すぐに実行できるのです。そのうえ，得意先などでの活用を考慮して通信機能も充実設計。音響カプラと組み合わせて電話機に接続すれば，データ通信も思いのまま。

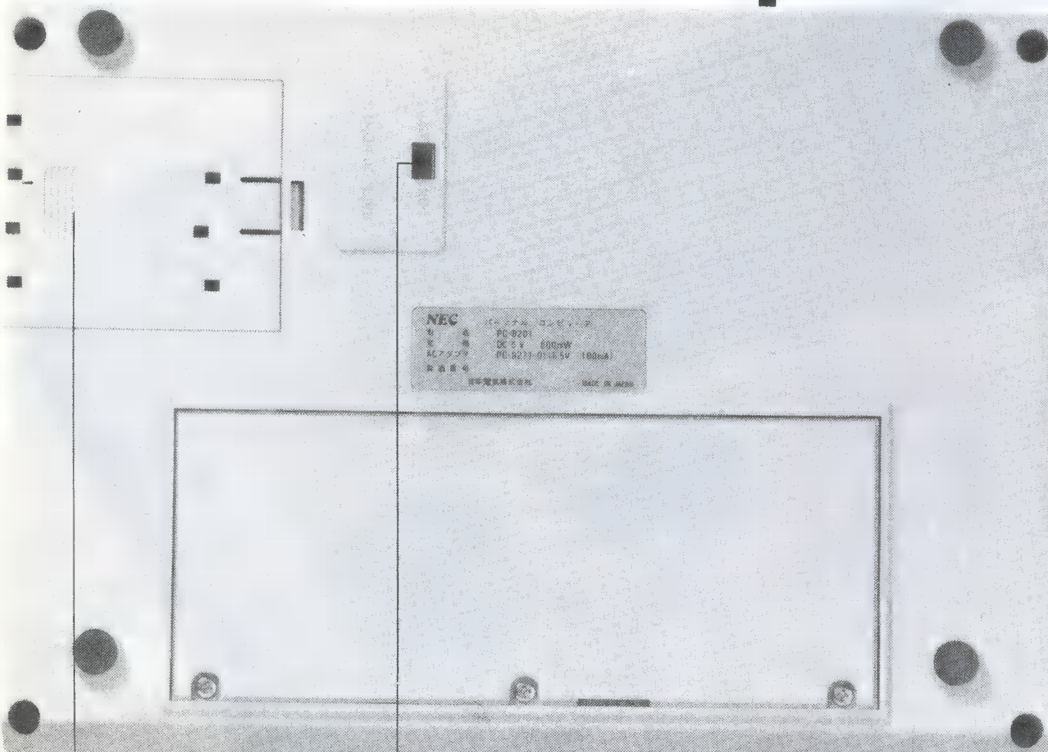
ビジネス用のハード機能はすべて備えているので，周辺機器を接続してシステムアップも可能です。PC-8201は，からだは小さくとも機能性バツグンの本格派マシンなのです。

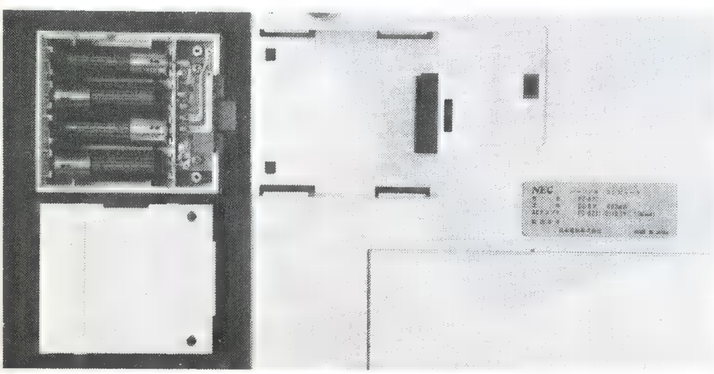
あなたもPC-8201でビジネスアップを。




ファンクションキー
液晶ディスプレイ
低電圧表示LED
インサートキー
デリートキー
NEC PC-8201
LOW BATTERY
カーソル
移動キー
コントラスト
調節ツマミ
電源スイッチ
リターンキー
シフトキー
SHIFT
スペースキー
カナキー







▲バッテリーカセットに単3乾電池4本をセット

AD・林夏介
編集協力・桂樹社グループ
図版・日進工房
イラスト・林夏介
カット・原田紘一

第1章

PC-8201の概要



1 PC-8201の特徴

PC-8201 の特徴を説明する前に，これからの話の前提となる3つの言葉，「プログラム」，「データ」，「メモリ」の概念について理解していただこうと思います。

「プログラム」とは，コンピュータにさせる仕事の手順を書いたものです。コンピュータに実際に仕事をさせる時は，このプログラムをコンピュータに覚え込ませてから行います。

「データ」とは，コンピュータで処理される数値や文字の情報のことで，そのつどタイプライタ等からタイプして伝えることもあれば，あらかじめ，まとめてタイプして保存しておくこともあります。データをたくさんまとめて保存したものを，特に，「ファイル」と呼びます。

「メモリ」とは，プログラムやデータを記憶しておくコンピュータ内の記憶装置のことです。コンピュータ外部からメモリへプログラムやデータを記憶させることを入力といい，メモリからコンピュータ外部へ取り出すことを出力と呼びます。

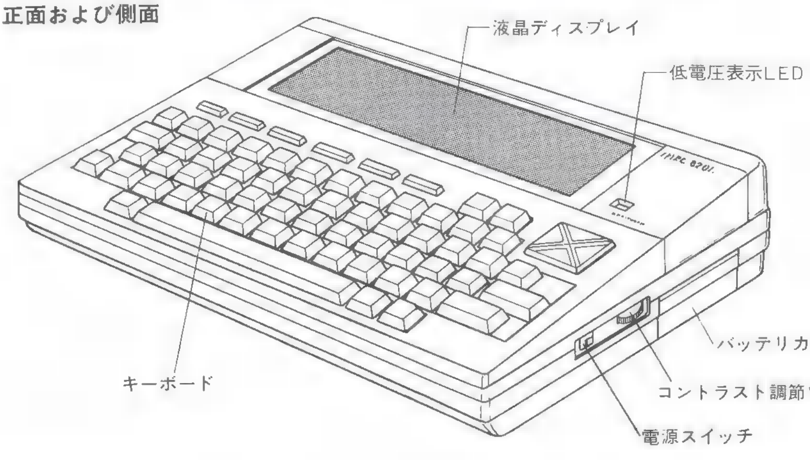
これらの用語については，先へ進んだ時，もう少し詳しく（専門的に）解説することになりますが，ここでは，この程度の紹介にしておきます。

さて，ハンドヘルド・パソコン「PC-8201」の第1番目の特徴は，ポータブルであるということです。これは，A4サイズで軽いということもありますが，小型というだけではありません。戸外でも車の中でも使えるようにと，乾電池で動くので，コンセントの心配がいらないのです。もちろん，アダプタコードで使うことも可能です。

さらに，通常のパソコンのように電源を切るとメモリ内が消えるのではなく，乾電池やコードなしでも，7～26日はメモリ内のプログラムが消えないのです。したがって一度メモリ内にプログラムを記憶させてしまうと，プログラムのロード（カセットテープやフロッピーディスクに入っているプログラムをメモリへ移すこと）をせずにいつでもプログラムを実行できます。

また，通常のパソコンでは，メモリには現在動かしている1本のプログラムと，そのデ

【図1】 正面および側面



ータしか記憶されませんから，現在のプログラム以外にとっておきたいプログラムがある時や，後で使いたい記録，つまりデータファイルを保存したい時は，カセットテープやフロッピーディスクへ別に記録します。PC-8201では，メモリ容量に余裕さえあれば，最大63までのプログラムやファイルを自分自身に記憶させることができます。言い換えますと，PC-8201では，カセットレコーダやフロッピーディスク装置を使ったと同じような処理が，本体だけで可能なのです。

PC-8201では，現在動かしているプログラムとデータを「BASIC領域上」のプログラムやデータと呼び，保存のために記憶されている方を「RAM上」のプログラムやファイルと呼びます。RAM上のプログラムはいつでも，BASIC領域に移して動かすことができます。

第2の特徴は，形はポータブルであってもビジネスパソコンに必要なハードウェア的な機能をすべて持っていることです。

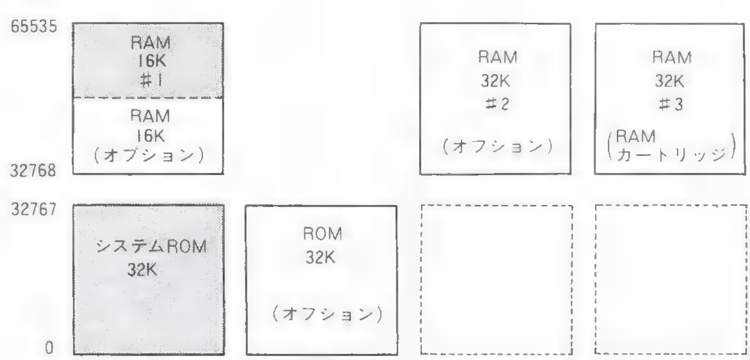
まず実物を見るとすぐわかることですが(図1)，キーボード（本体のタイプライタ部分）は，小型コンピュータにありがちな縮小サイズではなく，JIS規格どおりなので，スムーズに操作ができます。液晶ディスプレイは，横40字，縦8行，合計で320字まで表示できるので，プログラムの表示などにも問題なく使えます。本格的なディスプレイ装置が横80字，縦25行表示できるのに比較すると不十分ですが，これまでのハンドヘルド・コンピュータが20～80字程度しか表示できなかったことを思うと画期的といえます。

次にメモリの大きさですが，標準で16Kバイト RAMと，32Kバイト ROMを持っています。本体内で64Kバイト RAM，64Kバイト ROMまで拡張できるほか，カートリッジを使えば，32Kバイト RAMか，128Kバイト ROMのいずれかを追加できます（図2）。

ここでRAM，ROMの用語説明をしておきます。RAMとは，RANDOM ACCESS MEMORYの略で，自由に読み書きができるメモリの意味です。具体的には，プログラム等が記憶されるメモリで，ユーザーズメモリ（使用者用メモリ）とも呼ばれます。

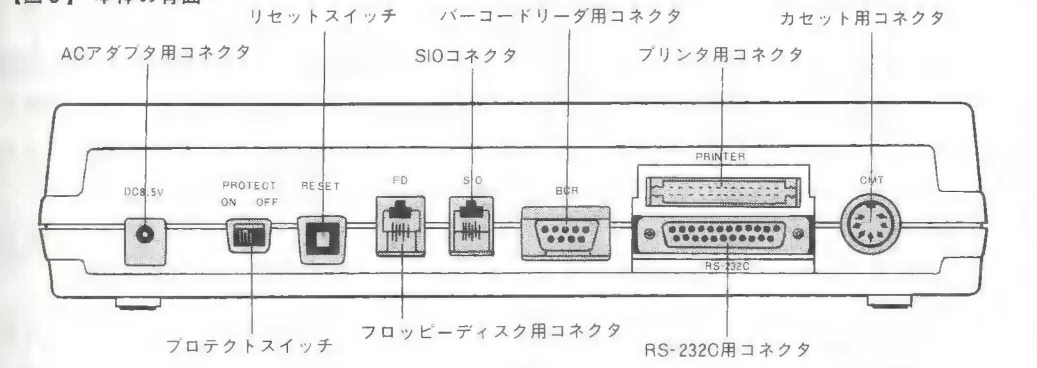
ROMとは，READ ONLY MEMORYの略で，読み込み専用メモリの意味です。ROMには，工場出荷時に特定の内容が記憶させられていて，いくらパソコンを使っても

【図2】 メモリマップ



●RAM#2，RAM#3のアドレスは0～32767または32768～65535のどちらにも指定できる。
●各ブロックは，32Kバイト単位でバンク切り換えを行うことができる

【図3】 本体の背面



内容が消えたり，変化したりはしません。ROMには，BASIC言語など，パソコンの機能そのものが内蔵されています。

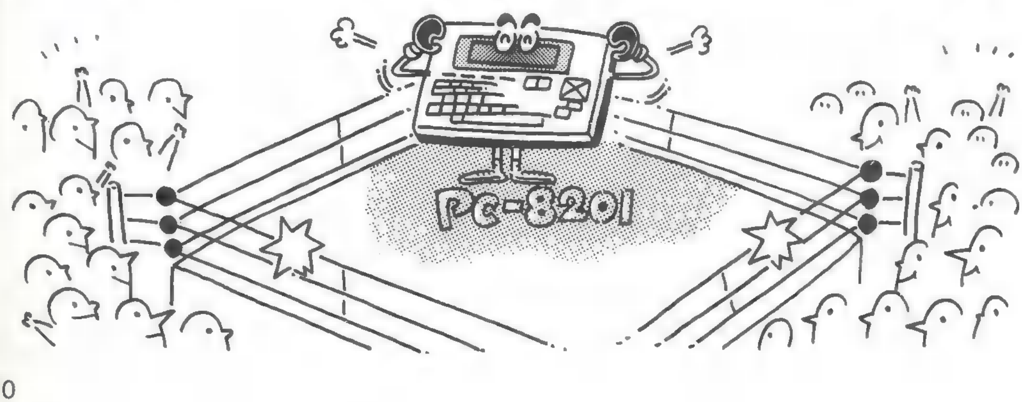
Kバイトといってもピンとこない方もいるかもしれません。バイト（BYTE）とは，コンピュータにおける表現の単位で，1バイトがアルファベットやカナ文字の1文字を表すと考えてください。K（ケイ）はキロとほぼ同じ意味です。したがって，16Kバイトとは1万6000字分のメモリです。パソコンの代表的な機種PC-8001のメモリが，標準で16KバイトRAMと24KバイトROM，拡張して32KバイトRAMと32KバイトROMでしたから，PC-8201のメモリが十分というのがおわかりいただけると思います。

ハードウェア上の特徴としては，さらに，各種の周辺装置の接続を十分考慮した設計になっているということです。カセットテープレコーダ，プリンタ，バーコードリーダ，音響カプラや他のコンピュータと接続するためのRS-232Cの各インターフェイスを標準装備しているほか，ミニフロッピーディスク装置，CRTディスプレイ装置との接続もできるようになっています（図3）。

3番目には，PC-8001に内蔵されているN-BASICと，ほぼ同じN_{82}-BASICを内蔵しているため，豊富なN-BASIC用のプログラムの大部分を，若干の手直しですぐ使えること，およびBASICと同等の形で，TEXTと，TELCOMの2つのソフトウェアを内蔵していることがあげられます。

この2つのソフトウェアについては後の章で詳しく述べますが，TEXTは文章編集機能，TELCOMはターミナル機能をそれぞれ持ち，単独で，あるいはBASICと関連して働きます。

さらには，BASIC，TEXT，TELCOM，ユーザーが作成したプログラムやデータファ



イル等，メモリ上にあるものはすべて自動的にメニュー（MENU）画面に表示され，プログラムの実行や印刷・消去などの処理がすべてメニュー画面上の選択で行うことができます。ユーザーは，コマンドやファイル名をタイプすることなく，ファイル管理を行えるのです。メニューについては操作の仕方で改めて説明します。

② PC-8201の操作

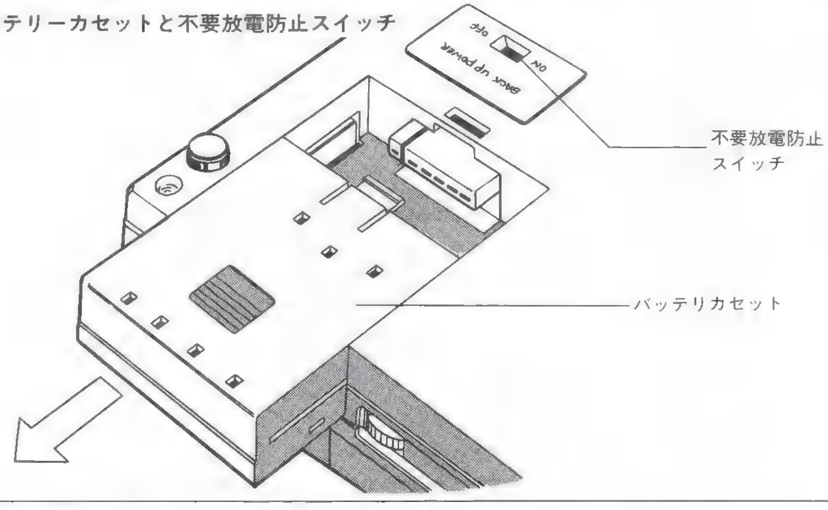
電源の入れ方

バッテリカセットに単3乾電池を4本入れて本体に装着するか，ACアダプタにより，コンセントにつないでから，PC-8201裏面にある不要放電防止スイッチをONにし，電源スイッチをONにします（図4）。

パソコンを使い終わった時は，電源スイッチのみをOFFにして，不要放電防止スイッチはそのままにしておきます。このスイッチがONの時は，電源スイッチがOFFの間，非常用Ni-Cdバッテリを働かせ，RAMの内容が消えないように保存します。このスイッチをOFFにすると，バッテリの放電は止まりますが，RAMの内容が消えてしまいます。通常の使用頻度の時は不要放電防止スイッチをONのままにしておき，非常用バッテリが完全放電してしまうぐらい長期間使用されない時は，RAM上のファイルをカセットなどに移したうえで，OFFにしておくとよいでしょう。

また，PC-8201には，10分以上キー入力がない場合に，自動的に電源をOFFにする機能があり，オートパワーオフ機能といいます。これはバッテリの消耗を防ぐためで，もちろん，キー入力がなくても，TELCOMモードで通信中の場合や，計算や印刷が実行中のBASICプログラムの場合には，オフになりません。オートパワーオフ機能で電源オフになった後，もう一度電源をオンにする時は，電源スイッチをOFFにしてから入れ直します。BASICのコマンドレベルにあった場合はメニューへ，その他の場合は元の画面へ戻ります。

【図4】 バッテリーカセットと不要放電防止スイッチ



メニュー（MENU）画面

電源スイッチが入れられると，PC-8201のディスプレイ画面は図5のようになります。画面の濃さがみづらい時は，電源スイッチの上方にあるコントラスト調節ツマミを前後に回して，調節できます。

この画面はメニューと呼ばれる画面で，料理の献立表と同じく，PC-8201でできる仕事内容，具体的にはプログラムとファイルの名前を表示しています。PC-8201には，あらかじめ，BASIC，TEXT，TELCOMの3つのソフトウェアが内蔵されていますが，ユーザーが自分のプログラムをPC-8201の空き領域に書き込んでいけば，ユーザーのプログラム名もまた，メニュー上に表示されます。データファイルを書き込んだ場合も同様に，メニュー上に表示されます。これらのプログラムやデータファイルを扱う時は，すべてメニュー画面上で行うことができます。

現在，画面上の「BASIC」の文字の部分だけが白黒反転した表示になっています。この反転している状態をリバースといい，メニュー上から選択されたプログラム，またはデータの名前の部分がリバースで表示されます。次の「TEXT」を選択したい時は ◁ キーを1回押します。リバース状態はBASICの文字からTEXTの文字へ移ります（図6）。

メニュー画面にたくさんのプログラム名が表示されている時は，▽ △ ▷ ◁ のキーを押して必要なプログラム名がリバースになるようにすればよいのです。

メニュー画面の左上に表示されている日付と時間は，普通のデジタル時計と同じで，一度正しい日付と時間をセットすれば，きちんと時を刻みます。プログラムなどと同じく，

【図5】

```
1983/01/01 00:00:00    (C) Microsoft #1
 BASIC       TEXT       TELCOM     -.-
 -.-         -.-        -.-        -.-
 -.-         -.-        -.-        -.-
 -.-         -.-        -.-        -.-
 -.-         -.-        -.-        -.-
 -.-         -.-        -.-        -.-
Load     Save     Name     List     28758
```

【図6】

```
1983/01/01 00:00:30    (C) Microsoft #1
 BASIC       TEXT       TELCOM     -.-
 -.-         -.-        -.-        -.-
 -.-         -.-        -.-        -.-
 -.-         -.-        -.-        -.-
 -.-         -.-        -.-        -.-
 -.-         -.-        -.-        -.-
Load     Save     Name     List     28758
```

【図7】

```
NEC PC-8201 BASIC Ver 1.0 (C) Microsoft
28758 Bytes free
Ok
■




Load "   Save "   Files   List     Run
```

電源を切っても消えないので，改めて日付や時間をセットする必要はないのです。

メニュー画面の右上の（C）Microsoftは，PC-8201のソフトウェアの版権がマイクロソフト社にあることを示しています。

右端の＃1は，現在使用しているRAMのグループの番号が1であることの表示です。PC-8201では，最大96Kバイトまで RAMを拡張できますが，それを32Kバイトずつのグループに分け，それぞれをバンク1，2，3と呼びます。同時には1つのバンクしか使えません。現在メニューで表示されているのは，バンク1の内容です。

メニュー画面の右下の数字は現在使用されているバンクの残っている領域の大きさ，つまりバイト数を表しています。

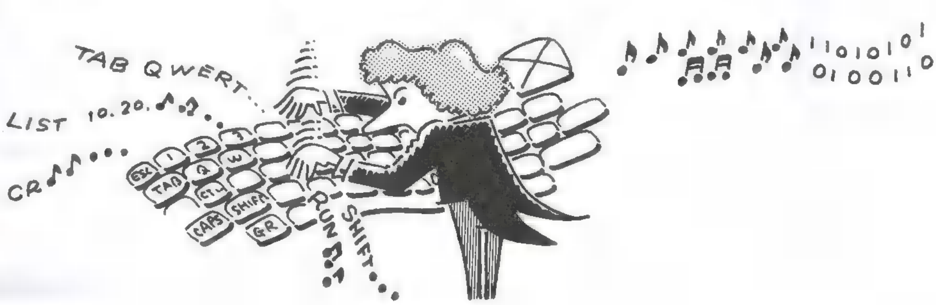
画面下に表示されている「Load」，「Save」などの単語は，キー操作のところで説明するファンクションキーと対応していて，メニュー画面が表示されている状態で，PC-8201に命令できる内容です。メニューモードにおけるコマンド（COMMAND）ともいいます。

では，もう一度「BASIC」の文字がリバース状態になるようにして，［⏎］マークのキーを押します。これで，メニューからBASICのソフトウェアを選択し，実行したわけです。PC-8201は，BASICモードになります（図7）。

BASICモードの最初は，必ず図のような画面から始まります。2行目にある数字は，BASICで使うことができるメモリの残りのバイト数です。新しいファイルをメモリに保存するごとにこの数は減っていきます。

キー操作

PC-8201の表面のタイプライタに似た部分をキーボードと呼び，カナ文字の一部を除けばJIS規格のタイプライタと同じ配列をしています（図8）。



キーボード上には，白っぽいキーと濃い色のキーがあります。白っぽいキーは，普通キーとか一般キーと呼び，押すとそのキーの表面に書かれている文字がコンピュータに伝えられます。色の濃いキーは特殊キーといい，文字ではなく，ある動作・機能をコンピュータに伝えるために用います。

さて，キーを順番に押してみましょう。沈んだままのキーがないか調べ，あれば1回押すと戻るので全部元へ戻してから，キーボードの1番上の段の数字のキーを押します。

1文字押すごとに，その文字が画面に表示され，画面上の黒い四角が右へずれていきます。この四角を「カーソル」と呼びます。現在カーソルは最後の文字の右にあります。カーソルは，「次はここに表示します」という文字の表示位置を示しています。

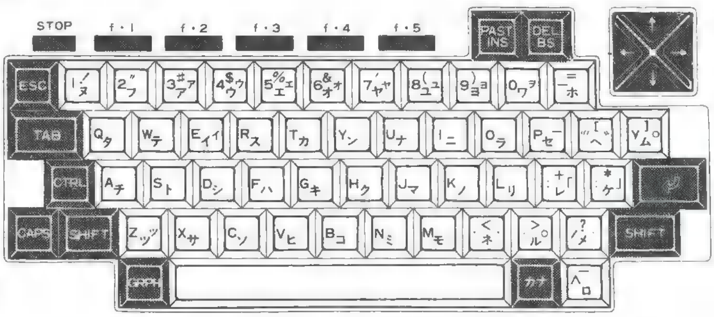
アルファベットも同様に押してみます（図9）。

1行は40文字まで表示できますが，それを超えると自動的に次の行に移ります。アルファベットは小文字で表示されますが，あなたのパソコンで大文字かカナが表示される時は，CAPS または カナ のキーが沈んでいないか調べてください。

CAPS キーは大文字ロックの働きを持っているため，このキーが沈んだままで，アルファベットのキーを押すと大文字表示になります。CAPS キーは，大文字でアルファベットを続けてタイプしたい時に使うもので，アルファベット以外の数字や記号には，押されていてもいなくても何の影響もありません。

SHIFT と書かれたキーが左右に1個ずつあります。これはシフトキーといい，このキーを押しながら，文字のキーを押すと，アルファベットの大文字，または記号がタイプできます。左右のシフトキーのどちらを押してもかまいません。

【図8】キーボード



【図9】

```
NEC PC-8201 BASIC Ver 1.0 (C) Microsoft
28758 Bytes free
Ok


1234567890-qwertyuiop@¥asdfghjkl;:zxcvbn
m,./■
Load "  Save "  Files   List     Run
```

[カナ]と書かれたキーは，カナ文字タイプのために使います。[カナ]キーを一度押すと，キーは沈んだままになり，文字のキーを押すとすべてカナ文字となります。[カナ]キーは，もう一度押すと，カナロックが解けます。[カナ]キーと[SHIFT]キーの両方を押して，文字キーを押すと，カナ記号がタイプできます。

[SHIFT]キーと[カナ]キーの関係は図10のようになります。

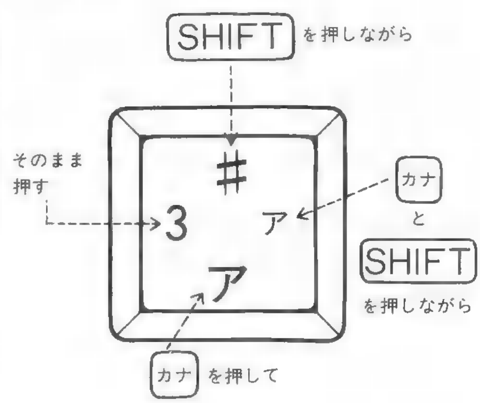
このほか，キーの表面に書かれていない記号をタイプすることができます。[GRPH]キーを押しながら[Z]キーを押すと，画面に◀の記号が表れます。[GRPH]キーはグラフィック（GRAPHIC）模様を表示するために使うものでグラフキーと呼びます。[GRPH]キーを押しながら，同様に[X]と[C]を押すと，それぞれ↵と▩が表示されます。[Z]，[X]，[C]以外のキーには，最初はグラフィック記号は用意されていませんが，ユーザーが61種類までグラフィック記号を定義することができます。

[カナ]キーの左側にある何も書かれていない細長いキーはスペースバーといい，空白をタイプする時に使います。このスペースバーを押すと，カーソルは右へ1つ移動します。画面には何も表れませんが，コンピュータ内では1文字分として数えられます。

文字のキー全部と，一部の特殊キーは，1秒以上押し続けると，同じ文字がたくさん表れます。これを「リピート機能」といいます。同じ文字を何度も続けて押す時には便利ですが，キーボードに慣れないうちは強く押してしまいがちで，1文字のつもりで何文字か続けてタイプされるので注意してください。

これで，一般の文字キーは，すべて使ってタイプできるはずです。キーの配列は一見デタラメなようですが，使用頻度の高いキーを指が動きやすい順に並べてあります。自分の名前や住所をアルファベットやカナを使っていろいろタイプし，キーの配列に早く慣れてください。

【図10】 キーの文字と記号



次にタイプしていて，タイプミスをした時はどうすればよいかを説明します。

例えば，「ABCDE」とタイプするつもりで「ABVDE」とタイプしたとします。カーソルはEの右側にあるはずです。[◁]キーを3回押して，誤った文字Vに戻します。そこで正しい文字[C]をタイプすれば画面の文字はVからCへ変わります。

ABVDE■
↓
AB**V**DE
↓
ABC**D**E

余分な文字をタイプした時は，[DEL BS]のキーを使って削除します。[DEL BS]キーを1度押すとカーソルの前の文字が，1つ削除されてカーソルから後が全体に前へ移動します。つまり，デリート（DELETE）の働きをするキーなのです。

1 2 3 4 5 6 **7** 8 9 0 −
↓ [DEL BS]キーを押す
1 2 3 4 5 **7** 8 9 0 −

[SHIFT] キーと一緒にこの [DEL/BS] キーを押すと，カーソルのある文字が消え，カーソルより後が全体に前へ移動します。

1 2 3 4 5 **7** 8 9 0 −
↓ [SHIFT] と [DEL/BS] キーを押す
1 2 3 4 5 **8** 9 0 −

カーソルを中心に後ろ側を削除していくので，バックスペース（BACK SPACE）の働きを持っています。

何文字かを抜かしてタイプしてしまった時は，[PAST/INS] キーを使って挿入ができます。インサートキーと呼びます。[PAST/INS] キーを1回押すと，カーソルは■から＿（アンダーライン）に変わります。カーソルがアンダーラインの時は，カーソルの前に文字が挿入できます。6，7の文字をタイプすると，カーソルより後が全体に後ろへ移動し，カーソルの前に6，7の文字が表示されます。

1 2 3 4 5 <u>8</u> 9 0 −
↓
1 2 3 4 5 6 7 <u>8</u> 9 0 −

[PAST/INS] キーをもう1回押すと，カーソルは元の■に戻り，挿入状態は解除されます。◁ などでカーソルを移動したり，[⏎] キーを使った時にも挿入状態は解除されます。

[SHIFT] キーを押しながら，[PAST/INS] キーを押すと，PASTE バッファの内容が画面に表示されますが，これは第3章の「TEXT」で説明します。

カーソル移動の4つのキー ▽ △ ◁ ▷ および [DEL/BS] [PAST/INS] のキーを使うと，このように画面上の文字を簡単に修正できます。パソコンに情報を伝えるためのもっとも一般的な方法は，キーボードをタイプすることなので，パソコンを活用する第一歩はキーボードに早く慣れることです。

PC-8201 のキーボードは，ポータブル用でありながら標準タイプライタと同じ大きさなので，少し練習すれば，指が自然と動くようになるはずです。

コンピュータに情報を伝えることを「入力」または「インプット」（INPUT）といいます。タイプライタから文字をタイプするのは，もちろん，この入力のためですが，画面に表示しただけでは入力にはなりません。タイプが終わったところで，区切りとして [⏎] キーを押します。[⏎] はリターンキーといい，タイプされた内容をコンピュータに伝える働きと，次の行の最初へカーソルを戻す働きがあります。画面にでたらめな数字をタイプして [⏎] キーを押してみましょう。

[⏎] キーを押したことにより，画面上の「1234567890-」がコンピュータに伝えられ，BASIC は，この文字の意味を判断しようとします。BASIC の文法に沿った命令ならば，機械用の2進数の命令に通訳され，直ちに実

【図11】

```
1234567890-
?SN Error
Ok
■




Load "  Save "  Files  List    Run
```

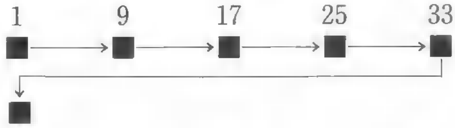
行されますが，この数字は，BASIC の命令ではなかったため，文法的におかしいの意味をこめて「?SN Error」のメッセージが表示されます。「Ok」の文字は正しいの意味ではなく，1つの仕事が完了したので，次の命令待ち状態であることを表しています（図11)。

このように，[⏎] キーはコンピュータを働かせるためのキーです。具体的には，命令をタイプした時の終わり，プログラムの1行の終わり，コンピュータ側から入力せよの指示があってデータをタイプした時の終わりに押します。TEXT では単に改行します。

[⏎] キーを押して，PC-8201 に何らかの仕事をさせている時に，その仕事を中止させたい時は，キーボードの左上にある [STOP] キーを押します。このキーは，その名のとおり，ストップさせるキーです。仕事が PC-8201 だけでなく，カセットテープレコーダやプリンタと一緒の時は，[SHIFT] キーと [STOP] キーを同時に押して，ストップさせます。キー操作をしているだけでは，特に使うこともありませんが，プログラムを実行するようになると，必要になるキーです。

では，左上の [ESC] キーを説明します。このキーは，エスケープキーと呼ばれ，続けて他のキーを押すことで，特殊な機能を果たします（エスケープキーと他のキーの組合せについては，161ページ参照)。

[ESC] キーの下にある [TAB] キーは，タブキーといい，カーソルを8桁単位で右へ移動させます。[TAB] キーを5回押してみます。カーソルは次の行の最初まで移動します。



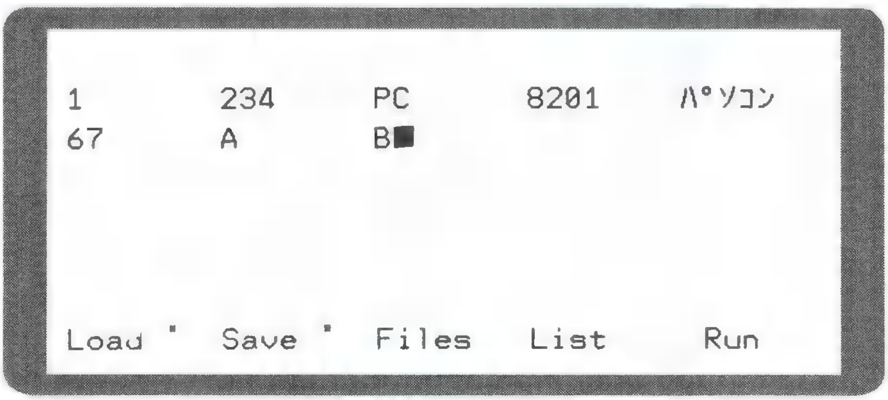
今度は，文字をタイプしながら [TAB] キーを押してみます（図12)。1 [TAB] 2 3 4 [TAB] PC [TAB] 8201 [TAB] パソコン [TAB] ……。

[TAB] を押すと，きちんと桁ぞろえして入力ができます。BASIC における [TAB] はスペースバーを8回押したと同じですが，TEXT ではさらに追加の機能があります。それについては第3章で述べます。

[TAB] キーの下の [CTRL] はコントロール（CONTROL）キーといい，他のキーと一緒に使うと特殊な機能の入力として働きます。[CTRL] キーと一緒に [L] を押してみます。画面がすべて消去され，カーソルは左上へ移動します。[CTRL] キーと一般キーとの組み合わせは，BASIC，TEXT，TELCOM のそれぞれで異なるので，ここでは，BASIC の場合のみ表にしておきます（表1)。

最後に残ったキーボードの一番上にあるキーをファンクションキーといいます。他のキ

【図12】



ーと同じく，文字のタイプや機能を働かせるために押すキーですが，働きは固定的ではなく，命令で自由に変えられます。各キーの働きは，そのつど違うので，現在登録されている機能が，画面の一番下の行に表示されます(図13)。

f·1 をファンクションキーの1番と呼び，使い始めでは「Load "」の文字が登録されていて，このキーを押すと「Load "」の文字が画面表示されます。キーボードから「Load "」の文字をタイプしたのと同じ働きなので，タイプのスピードアップになります。 f·1 から f·5 までを順に押してみます。「?SN Error」になりますが，今は気にしないでおきましょう(図14)。

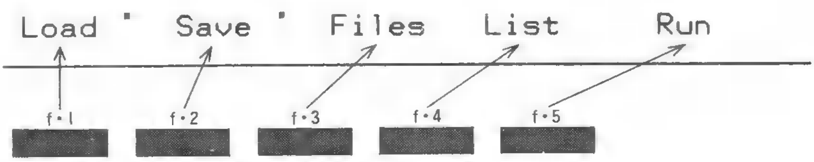
SHIFT キーを押しながら，画面の一番下に表示されているファンクションキーの内容を見ると，図15のようになっています。 SHIFT キーを押した時は，一般キーと同じように別の機能を持っています。

SHIFT キーと f·1 を f·6 ， SHIFT キーと f·2 を f·7 とそれぞれ呼びます。つまり，ファンクションキーは物理的には5

【表1】 コントロールコード表

CTRL + C	STOP と同じ
CTRL + E	カーソル以下を削除
CTRL + H	DEL BS と同じ
CTRL + I	TAB と同じ
CTRL + K	カーソルをホームポジション(画面左上隅)に移動する
CTRL + L	画面を消去し(プログラムは消えない)カーソルをホームポジションに移動する
CTRL + M	⏎ と同じ
CTRL + Q	CTRL + S を解除する
CTRL + R	PAST INS と同じ
CTRL + S	プログラムをLIST中，画面のスクロールを一時停止する
CTRL + U	1行を画面から消去。プログラムの行は消えない

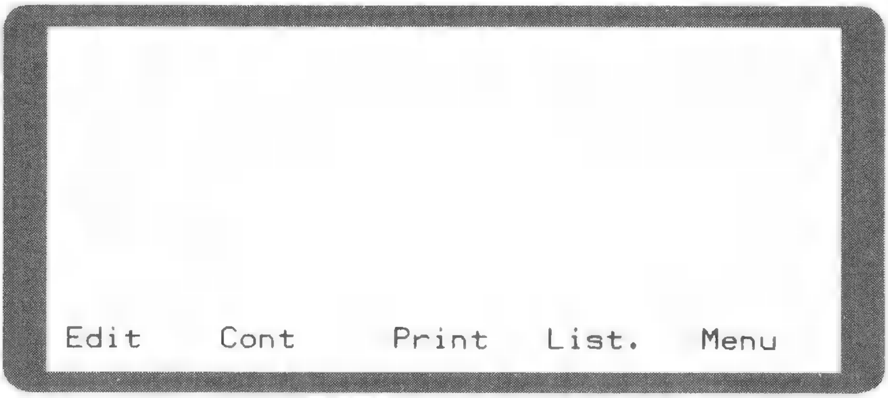
【図13】 ファンクションキーと画面下の表示との対応



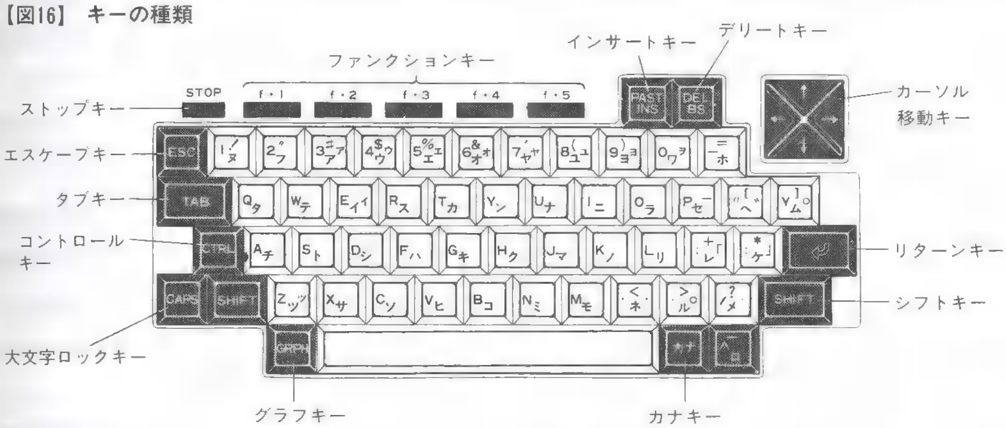
【図14】

```
Load "Save "Files
?SN Error
Ok
List Run
?SN Error
Ok
■
Load "  Save "  Files  List   Run
```

【図15】

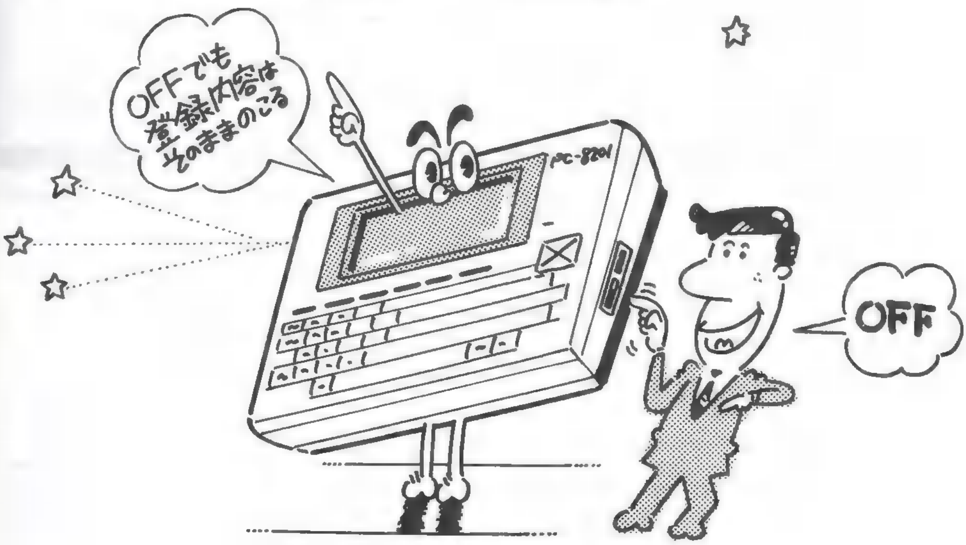


【図16】 キーの種類



個ですが，機能的には10個として働きます。

いま，ファンクションキーを10番目まで押すと，最後の「Menu」が働いて，メニュー画面へと戻ります。ファンクションキーの登録内容は使用者が変えることができ，一度変えると，新しい内容を登録しない限り，電源を切ってもそのままの内容が残るようになっています。



メニューの機能

PC-8201 の電源を ON にすると，通常，図17のメニュー画面が表示されます。この時 PC-8201 はメニューモードにあるといいます。

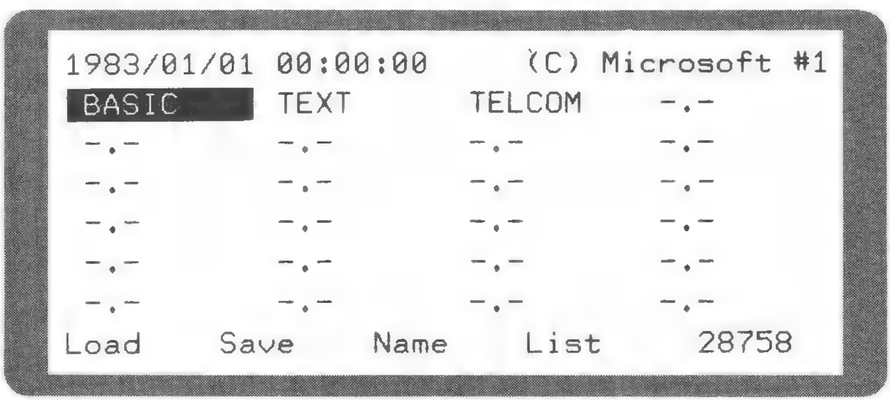
この PC-8201 には，BASIC，TEXT，TELCOM の3つのソフトウェアが内蔵されていて，それぞれが選択されている状態を BASIC モード，TEXT モード，TELCOM モードといいます。各モードの選択は，BASIC の EDIT 命令を例外にすると，必ずメニューモードから行います。メニュー画面上で各ソフトウェアの上へカーソルを移動して［⏎］キーを押すと，そのソフトウェアのモードになり実行されます。各モードからメニューモードに戻るためには，いずれもファンクションキー10番を押します（図18)。

メニュー画面には，BASIC，TEXT，TELCOM のソフトウェア以外に，ユーザーが作成し保存しているプログラムやデータファイルも表示されます。このユーザーが作成し保存しているプログラムやデータファイルを「RAM 上のファイル」と呼びます。

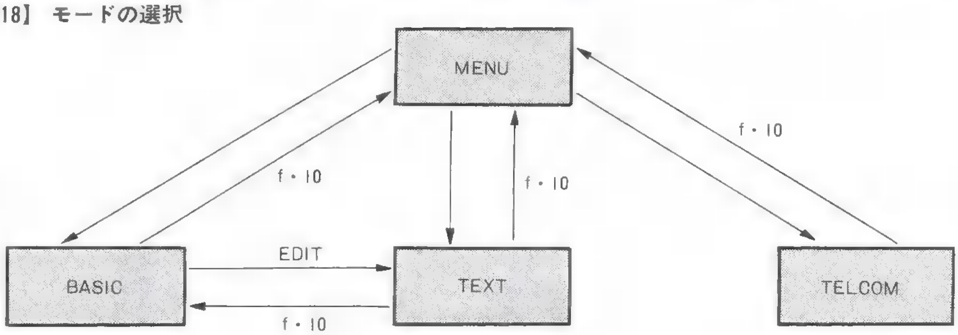
ファイルの名前は6文字以内でユーザーが自由に決められますが，名前の後には拡張子と呼ばれる2文字のアルファベットがピリオド（.）付きで自動的につけられます。拡張子が「.BA」のものを BASIC ファイル，「.DO」のものを TEXT ファイル，「.CO」のものを機械語ファイルといいます。

「.BA」の付いた BASIC ファイルは，バイナリー形式の BASIC プログラムで，通常の BASIC プログラムを RAM に保存した時には，すべてこの「.BA」ファイルになります。

【図17】



【図18】 モードの選択



【図19】

```
1983/01/01 00:01:30      (C) Microsoft #1
 BASIC      TEXT       TELCOM     TEST.DO
 ADD.DO     INSATU.BA  EX-1.BA    -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
Load     Save     Name     List      9745
```

「.DO」のTEXTファイルは，TEXTモードで作成したもの，BASICモードで作成したデータファイル，アスキーの形式のBASICプログラムが含まれます（図19）。

「.CO」の機械語ファイルは名前のとおり機械語で書かれたプログラムや，メモリの内容をそのままファイルに取り出したものです。

RAM上のファイルも，ソフトウェアを選択すると同じように必要なファイルの名前の上にカーソルを移動して選択できます。カーソルが重なったところで［⏎］キーを押すと，ファイルがBASICファイルならば，BASICモードに入ってそのプログラムが実行されます。ファイルがTEXTファイルならばTEXTモードになり，その内容が表示されます。ファイルが機械語ファイルならば，その機械語プログラムをメモリに読み込み，実行アドレスが指定されていれば実行に移ります。

BASICファイル，TEXTファイルを選択した場合は，それぞれ，BASICモード，TEXTモードに入っていますから，メニューモードに戻る時は，ファンクションキー10番を押します。機械語ファイルからメニューモードに戻る時は［⏎］キーを押してください。

そのほかに，ファイル名にカーソルを重ねてファンクションキーを押すと，ファイルに対する操作ができます。メニューモードで各ファンクションキーに登録されている機能を

【表2】 メニューモードにおけるファンクションキー

キー番号	キー	内容	機能
1	[f·1]	LOAD	外部装置からRAMへファイルを呼び込む
2	[f·2]	SAVE	RAM上のファイルを外部記憶へ書き込む
3	[f·3]	NAME	RAM上のファイルの名前を変更
4	[f·4]	LIST	RAM上のファイルを印刷
5	[f·5]	残りのメモリ	表示のみ
6	[SHIFT] + [f·1]	SET IPL	IPLコマンドファイルの指定
7	[SHIFT] + [f·2]	CLR IPL	IPLコマンドファイルの指定解除
8	[SHIFT] + [f·3]	——	未定義
9	[SHIFT] + [f·4]	KILL	ファイルの削除
10	[SHIFT] + [f·5]	BANK	バンク切替

表2に示します。

これらのファイル操作に必要な機能は，読んだだけではわかりづらいと思います。BASIC モードまたは TEXT モードで実際にファイルを作成した後，実習してください。

[f·1] に登録されている LOAD は，カセットやフロッピーディスク等の外部記憶装置からファイルを RAM に読み込む機能です。

[f·2] に登録されている SAVE は，RAM の内容を外部記憶装置へ書き込む機能です。第2章で説明する BASIC にも，LOAD と SAVE の命令があり，似たような働きをしますが，BASIC の LOAD, SAVE 命令が BASIC 領域内のプログラムを対象とするのに対し，メニューモードにおけるロード，セーブは RAM 上のすべてのファイル（プログラムも TEXT ファイルも）を対象とします。メニューモードにおけるロード，セーブについては，第2章 N_{82}-BASIC の中のカセットテープの入出力でも説明してありますので参照してください。

[f·3] に登録されている NAME は，RAM 上のファイルの名前の変更をします。名前を変えたいファイルへカーソルを移動して，[f·3] を押します。仮にそのファイル名が，GAME. BA とすると画面は図20のようになります。

「Name GAME. BA as—」に対して，新しいファイル名を入力します。この時，ファイル名に拡張子をつけることはできません。拡張子は現在と同じものが自動的につけられます。また，RAM 上にすでに存在しているファイル名と同じものを新しいファイル名として指定することもできません。入力ミスをすると，「ピッ」というブザーが鳴り，[f·3] を押す前に戻ります。

【図20】

```
1983/01/01 00:10:39      (C) Microsoft #1
 BASIC       TEXT        TELCOM      TEST.DO
 GAME.BA     -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
Name GAME.BA as _
```



【図21】

```
1983/01/01 00:10:55     (C) Microsoft #1
 BASIC      TEXT       TELCOM     TEST.DO
 TEST.BA    -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
Load     Save     Name     List      21450
```

新しい名前として[T][E][S][T]を入力します。[↵]キーが押されると，メニュー上の表示が直ちに変わります（図21）。

[f·4]に登録されているLISTは，RAM上のファイルを印刷するものです。第2章 N_{82}-BASICのプリンタによる出力を参照してください。

[f·5]に登録されているのは現在のバンクの空きバイト数です。[f·5]を押しても，ブザーが鳴るだけで，何もしません。

[f·6]に登録されているSETIPLと，[f·7]に登録されているCLRIPLは，イニシャル・プログラム・ローダー（INITIAL PROGRAM LOADER）コマンドに関するものです。

SETIPL（セットアイピーエル）は，電源が入れられた時に，自動的にさせる作業を設定し，CLRIPL（クリアアイピーエル）は，それを解除します。

SETIPLのためには，まず，TEXTモードで，電源が入れられた時，最初にさせたい仕事の命令群（これをコマンド列といいますが）を作成します。次にこのコマンド列を含むTEXTファイルへカーソルを移動して，[f·6]を押します（図22）。

SETIPLされたTEXTファイルは，他と区別がつくように「.DO」が「＊DO」に変わります。この「＊DO」を拡張子として持つTEXTファイルを，特にIPLコマンドファイルと呼びます。IPLコマンドファイルは，SETIPLをやり直すことで変更できます。別のTEXTファイルがIPLコマンドファイルに設定されると，元のIPLコマンドファイルの「＊DO」は「.DO」へ戻ります。

実際にIPLコマンドファイルを作成してみましょう。

カーソルをTEXTへ移し[↵]キーを押

【図22】

```
1983/01/01 00:11:02     (C) Microsoft #1
 BASIC      TEXT       TELCOM     TEST*DO
 TEST.BA    -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
Load     Save     Name     List      21450
```

します。モードはTEXTモードになります。

[I] [P] [L] と入力し，[⏎] キーを押します（図23）。

電源を入れた時に最初に実行させたい仕事がソフトウェアの選択であれば，BASIC, TEXT, TELCOM のいずれかを入力し，RAM上のファイルの選択であれば，そのファイル名を拡張子付きで入力します。

ここでは，TEST. BA というプログラムを選択することにします（図24）。

TEXT. BA の実行の開始時に，必ず何かを入力するのであれば，その値をコマンド列として続けておくこともできます。ここでは [1] を入力するとします。コマンド列は，最大64文字まで続けられます。ただし，[⏎] は2文字分と数えてください。コマンド列の入力が終わったらファンクションキー10番でメニューモードに戻します。IPL. DO という TEXT ファイルができています。カーソルを IPL. DO へ移し，[f·6] を押します。

【図23】

```
File to edit? IPL■
```

【図24】

```
TEST.BA←
1←
◀
```

【図25】

```
1983/01/01 00:11:08     (C) Microsoft #1
 BASIC      TEXT       TELCOM     TEST.DO
 IPL*DO     TEST.BA    -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
Load     Save     Name     List     21409
```

【図26】

```
1983/01/01 00:14:29     (C) Microsoft #1
 BASIC      TEXT       TELCOM     TEST.DO
 IPL*DO     TEST.BA    -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
Kill TEST.BA Sure?
```

IPL. DO は IPL＊DO に変わります（図25）。

それから一度電源スイッチを OFF にします。改めて ON にすると，IPL コマンドファイルが選択され，その内容が実行されて TEST. BA の実行に移ります。

PC-8201 は幅広い活用のできる汎用パソコンですが，一人ひとりのユーザーからみると，ある特定の業務を中心に使用することになるでしょう。その時，いちいち，メニューからプログラムを選択したり，初期値を入力するのは煩わしいことになります。IPL コマンドファイルを設定しておけば，いつでも自分の望むモードから自動的に開始することができます。SETIPL とは，オートスタート（自動開始）機能なのです。

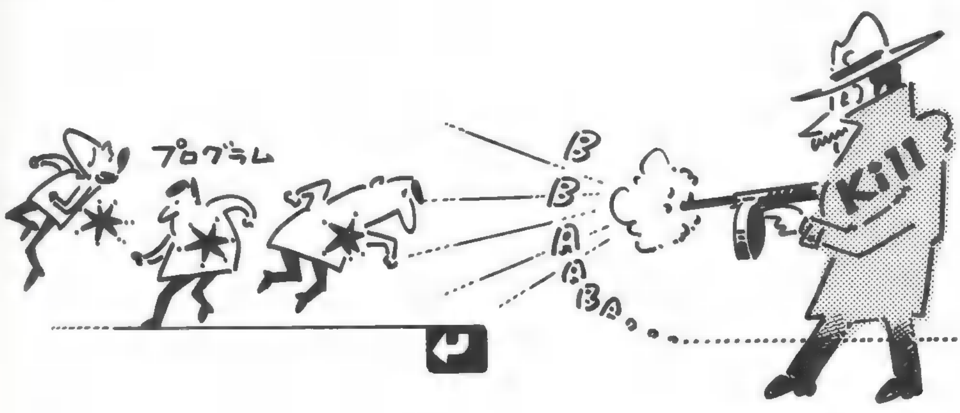
IPL コマンドファイルを解除するためには，現在の IPL コマンドファイルにカーソルを重ね，[f·7] を押します。

[f·9] に登録されている KILL は，RAM 上のファイルを消す働きをします。消したいファイルにカーソルを重ね，[f·9] を押します（図26）。

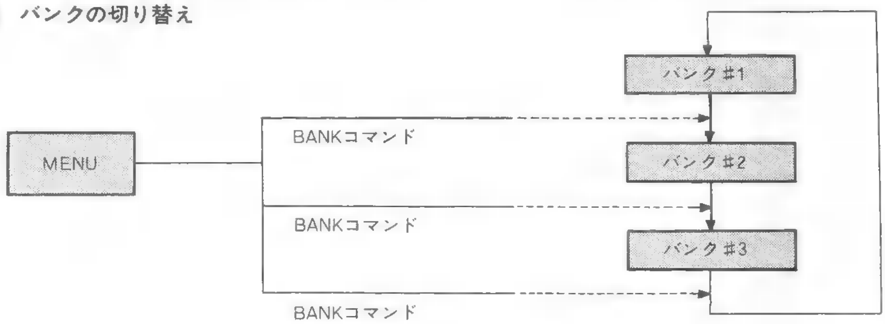
指定のファイルを KILL してもよいか (Sure) ? ときいてくるので，消してよければ [Y]，消したくない時は [Y] 以外を入力します。[⏎] キーはいりません。

[f·10] に登録されている BANK は，使用するバンクを切り換える働きをします。

PC-8201 では，最大 96 K バイトまでの RAM を増設することが可能ですが，32 K ずつのグループに分け，そのグループ単位に使用します。このメモリのグループが，バンクです。異なるバンクを同時に使うことはできませんから，PC-8201 で許されるファイルの大きさも 32 K が限度ということになります。



【図27】 バンクの切り替え



バンク1しかない時は，[f·10]を押しても何も働きません。バンク2，バンク3がある時は，[f·10]を押すごとに使用するバンクが切り換わります。現在どのバンクが使用されているかは，画面の右上に＃1，＃2，＃3の数字で表示されます。

また，メニューの内容も，バンク単位に表示されます。1つのバンクに記録できるファイル数は最大21ですが，バンクが3つある場合はその3倍，つまり，63まで可能です（図27)。

RAMの内容をすべて消してしまいたい時は，不要放電防止スイッチをOFFにすれば可能ですが，バンク単位に，RAMを初期化（工場出荷時の状態）したい場合は，コールドスタートという方法をとります。コールドスタートすると，BASIC領域のプログラム，メニュー画面に表示されているファイルはもちろん，日付，時間，ファンクションキーの内容，PASTEバッファの内容，文字領域の大きさ，使用するファイル数，通信形式など使用者が設定した項目はすべて初期状態に戻ります。

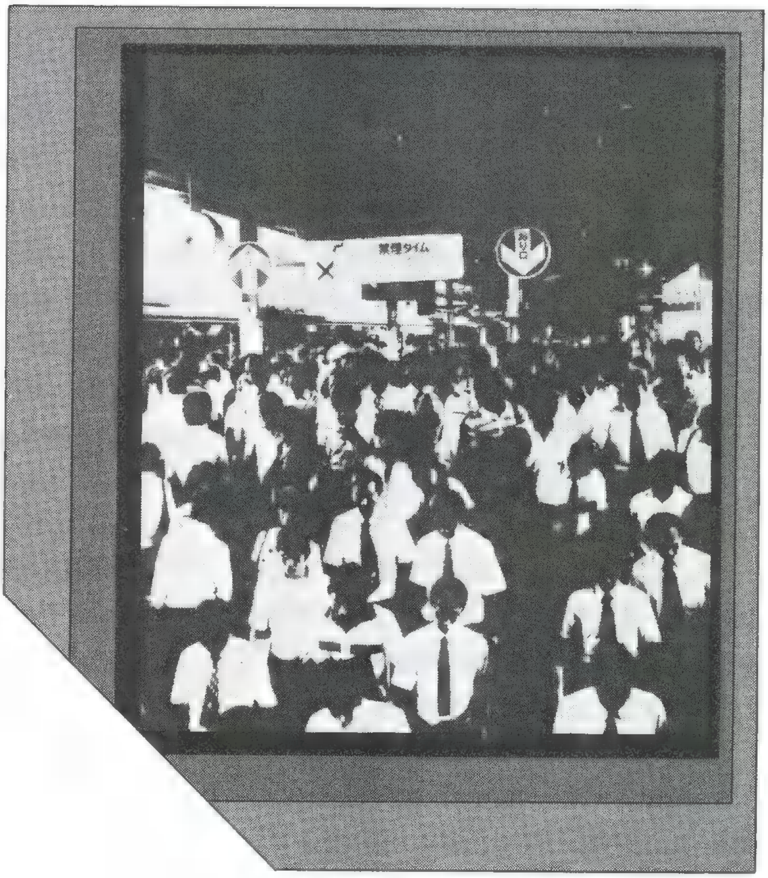
バンク1のコールドスタートは，[SHIFT]キーと[CTRL]キーを押しながら，リセットボタンを押します。

バンク2のコールドスタートは，バンク1で[f·10]を押し（バンク2へ切り換え），直後に[CTRL]キーを押します。

バンク3のコールドスタートは，バンク2で[f·10]を押し（バンク3へ切り換え），直後に[CTRL]キーを押します。

第2章

N82BASIC



1 BASICプログラムの基礎

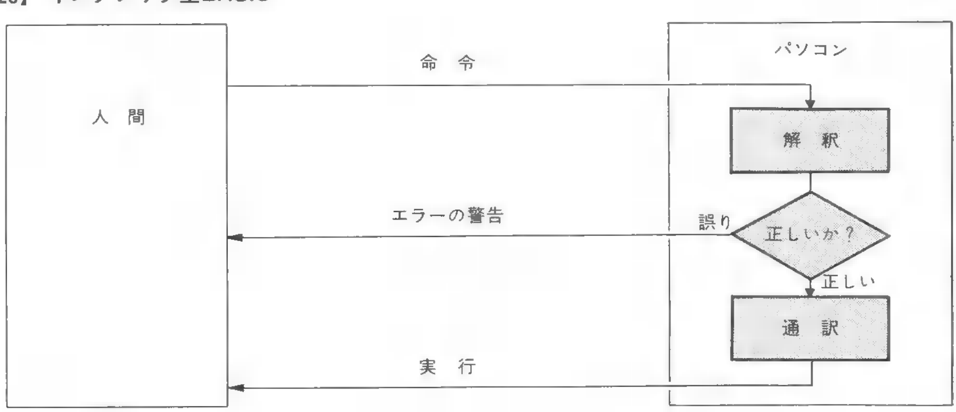
プログラムとは

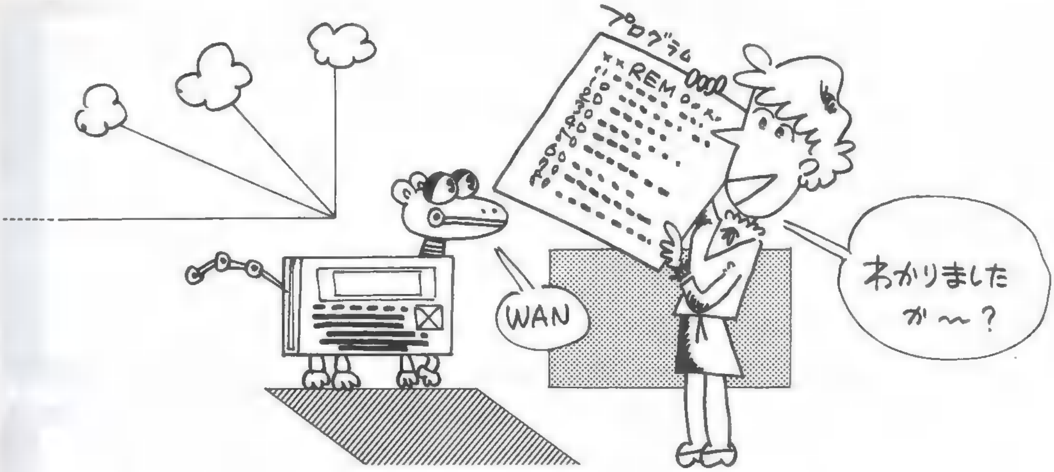
コンピュータにおけるプログラムとは，コンピュータにさせたい仕事の手順を，特定の規則に従って記述したものです。いいかえますと，プログラムとは，コンピュータに対するたくさんの命令の集まりです。プログラムを記述するための規則の体系のことをコンピュータ言語とか，プログラム言語と呼びます。コンピュータ言語には多くの種類があり，パソコンで使われているものだけでも十数種類ありますが，その誕生から現在に至るまでパソコンの言語といえば，BASIC が主流なのです。

プログラムが記述されると，コンピュータ言語はその文法に沿っているかどうかをチェックし，正しければ機械がわかる言葉（機械語またはマシン語といいます）に自分自身で翻訳または通訳します。コンピュータがプログラムを実行する時は，機械語でなければならないからです。そこで，直接機械語でプログラムすることもありますが，機械語は習得が難しく，でき上がったプログラムの修正も困難です。そのため，BASIC のような比較的やさしい言語でプログラムし，パソコン自身に内蔵されているBASIC 通訳プログラムに通訳させるのです。BASIC 通訳プログラムのことをBASIC インタプリタ，または単に BASIC と呼びます。

PC-8201 に内蔵されている BASIC は，特に N_{82}-BASIC と呼ばれ，パソコン界では神話に近いビル・ゲーツの率いるマイクロソフト社の開発によるものです。ひと口に BASIC といっても，各メーカー，各機種ごとに少しずつ違っているのが実情です。通常，これを「方言」と呼んでいます。なぜ方言が生まれるかというと，各パソコンのハードウェア上の機能の拡大に標準化が追いつかないためなのです。つまり，新しいパソコンに新しい機能が付加された時，従来の BASIC にその機能を使うための命令がなければ独自の新しい命令が追加されることになります。逆に機能が小さいパソコンでは，同じ命令があっても内容的に一部の機能しか果たせないこ

【図28】 インタプリタ型BASIC





とがあります。

PC-8201 の N_{82}-BASIC は，NEC の他の機種の BASIC とまったく同じではないのですが，ベストセラー機の PC-8001 に内蔵されている（そして，PC-8801 にも内蔵されています）N-BASIC とほぼ互換性を持っていますので，PC シリーズのプログラムは若干の手直しで簡単に移植できます。また，N_{82}-BASIC には，PC-8201 の特徴である「TEXT」や「TELCOM」ソフトウェアと関連する命令が新たに追加されています。

内蔵型 BASIC は，すでに述べたように「インタプリタ（INTERPRETER）」型 BASIC と呼ばれます。インタプリタ型言語では，命令を一つずつ機械語に通訳しながら実行します。別な言い方をしますと，命令を実行する時に，一つずつ命令の内容を解釈して正しければ実行し，正しくない時は，その場で警告メッセージを出して止まります。そのため，操作している人と，パソコンとが1対1で会話しているように命令と実行が行われます（図28）。

では，BASIC の命令を実際に使ってみましょう。まず PC-8201 を BASIC モードにします。メニュー画面で，[↑] [↓] [←] [→] のキーを使って BASIC の文字にカーソルを重ねます（図29）。[↵] キーを押すと，PC-8201 に記憶されているソフトウェアのうちの BASIC が選択された状態，つまり BASIC モードになります（図30）。

ここで，BASIC の中でもっともよく使われる PRINT という命令を使って，パソコンと会話してみます。

次のようにタイプしてください。

PRINT "abc" [↵]

「PRINT」は，プリントと読み，後に続く文字または数字を画面表示しなさいという命令です。PRINT の文字は大文字でも小文字でもよく，文字を表示する場合は引用符（"）で囲みます（図31）。

命令をまちがうと，BASIC は通訳して実行することができないため，警告を送ってきます。ためしに，次のようにタイプしてください。

PLINT "xyz" [↵]

SN ERROR は，「SYNTAX（文法）の誤り」の略で，もちろんエラーのある時は，命令は実行されません。「Ok」の文字は，正しいの意味ではなく「PLINT」命令の処理が終わったのでパソコンが何も仕事をしていない状態になり，次の命令を待っていることを表しています（図32）。

【図29】

```
1983/05/01 10:01:42     (C) Microsoft #1
 BASIC       TEXT      TELCOM      -.-
 -.-         -.-       -.-         -.-
 -.-         -.-       -.-         -.-
 -.-         -.-       -.-         -.-
 -.-         -.-       -.-         -.-
 -.-         -.-       -.-         -.-
Load     Save     Name     List     28758
```

【図30】

```
NEC PC-8201 BASIC Ver 1.0 (C) Microsoft
28758 Bytes free
Ok
■




Load "   Save "   Files   List     Run
```

BASIC の命令をタイプする時は，[⏎]キーで一区切りとされることを忘れないでください。[⏎]キーを押さずにスペースキーで次の行に移った時は，２行合わせて１つの命令とみなされます。

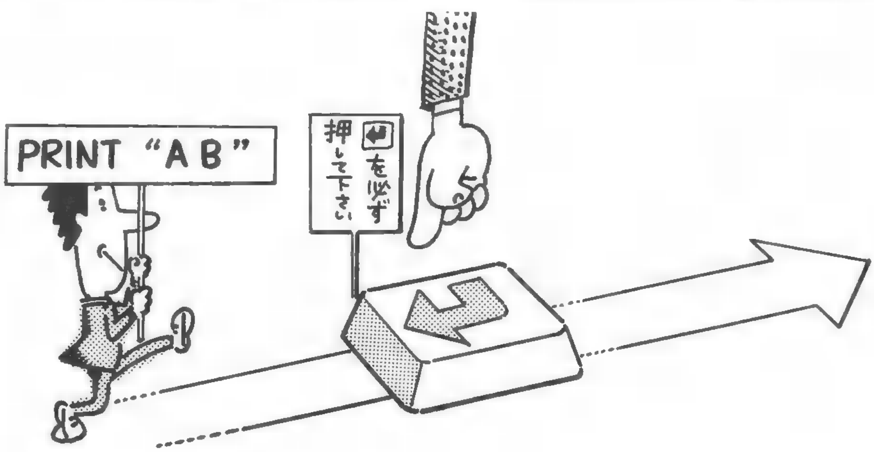
PRINT "AB"
PRINT "CD"[⏎]

この２行の PRINT 命令は，BASIC に PRINT "AB"PRINT "CD" と伝わり，文法的にエラーとなります。

また，同じ行に命令とは関係のない文字がある時，その文字も命令の一部とみなされます。

OK PRINT "X"[⏎]
PRINT "Y"[⏎]　12345

行のどこで[⏎]キーを押したかは関係がありません。[⏎]キーを押した時，同じ行にある文字すべてが１つの単位で BASIC へ



【図31】

```
NEC PC-8201 BASIC Ver 1.0 (C) Microsoft
28758 Bytes free
Ok
print "abc"
abc
Ok
■
Load "   Save "   Files   List     Run
```

【図32】

```
print "abc"
abc
Ok
plint "xyz"
?SN Error
Ok
■
Load "   Save "   Files   List     Run
```

伝えられます。

このようなミスを防ぐためには，画面を消してからタイプするとよいのです。画面全体を消すためには，[C] [L] [S] [⏎] と入力するか，[CTRL] キーを押しながら [L] を押します。

BASICの基本文法

PRINT 命令の例のように，BASIC では命令を1個ずつ入力して，1個ずつ実行させることもできますが，大量のデータを処理したい時などは，パソコンがいくら高速に処理しようにも命令のタイプに時間がかかって非常に効率が悪くなります。そこで，たくさんの命令をまとめてタイプし，それをパソコンに記憶させておけば命令のタイプは1回だけですみます。つまりプログラムです。プログラムでは，たくさんの命令が自動的に連続して実行されます。

命令が1個しかないときは，その命令を実行して終わりですが，たくさんの命令の集まりであるプログラムでは，どの命令を先に実行するかという順番が必要となります。BASIC では，この順番を行番号と呼び，命令の最初にタイプします。行番号は，実行の順番を表すものですから，同じプログラム内で同じ番号をつけることはできません。

「行番号」の規則

①0～65529の範囲内の整数であれば番号が飛んでいてもよい。

②同一プログラム内で同じ行番号をつけると，後にタイプした方が残る。

③1つの行番号に，続けて複数の命令を書くことができる。この時は各命令をコロン（：）で区切る(**図33**)。

④行番号から [⏎] キーまでを BASIC 上の

1行と呼び，255文字以内であれば，いくつの命令が含まれても，また画面上の数行に渡ってもよい。

では，プログラムを実際に作りながら，プログラムの作成・実行・修正の手順について説明しましょう。

図34の命令を順に入力してください。1行入力したら，必ず [⏎] キーを押します。これ以降のプログラムの説明では，[⏎] キーを押すことはいちいち書きませんが，特にことわりがなければ，行の最後は必ず [⏎] です。

行番号のない単独の命令の時は [⏎] キーを押すと直ちに実行され，命令が終わるとOKの文字が表示されますが，行番号付きでタイプすると，何行タイプしても実行されません。カーソルが次の行に移るだけです。

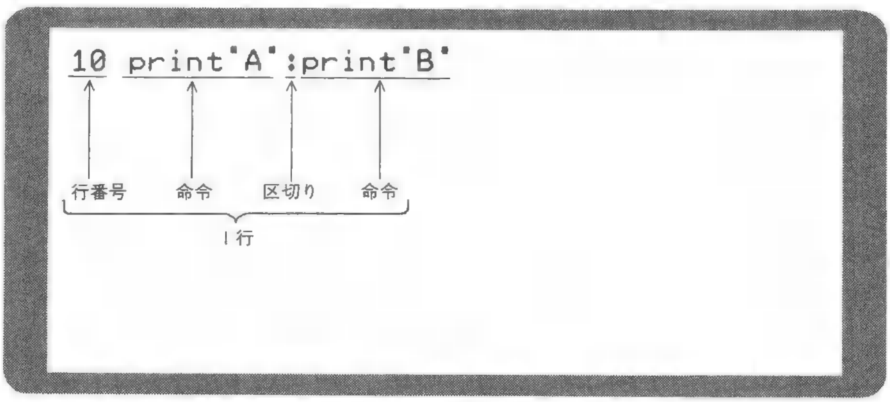
BASICは，0～65529の整数で始まるアルファベットの行をプログラムとみなし，実行せずにそのまま記憶します。まちがった命令でも，とにかく，そのまま記憶します。

記憶してあるプログラムを実行させるには，「Run」（ラン）という命令を使います。RUN [⏎] で，行番号の順に次々に命令を実行します（図35）。RUN [⏎] は [f·5] に登録されています。

プログラムの命令は，タイプした順に関係なく，行番号順に実行されます。したがって，後からいくらでも命令を追加できます。追加しやすくするために，通常，行番号を10番置きにつけます。

現在メモリ内にどのようなプログラムが記憶されているかをチェックするためには，「List」（リスト）命令を使います。LIST [⏎] で，行番号の順に命令が画面表示されます。画面一杯になると，スクロール（画面が1行ずつ繰り上がる）が起きます。LISTは [f·4] に登録されています（図36）。

【図33】



【図34】

```
1 print "PC-8201ハ"
3 print "ホ゜ータフ゛ルナ"
4 print "コンヒ゜ュータテ゛ス"
2 print "A4サイス゛ノ"
■

Load "   Save "   Files   List   Run
```

小文字でタイプしたprintの文字は，すべて大文字に変わっています。命令は，大文字，小文字のどちらでタイプしても，大文字に変換されます。ただし，(“ ”）の中は，タイプのままで変化しません。

プログラムの1行は，文字をタイプして画面に表示しただけでは記憶されず，［⏎］キーを押してはじめてメモリ内に記憶されます。同様に，メモリ内のプログラムは，画面から消えても，メモリ内にはそのまま残っています。もう一度ランさせることもリストさせることもできます。プログラムをメモリから消すには，命令が必要です。プログラム内の特定の行を消すためには，消したい行の番号をタイプし，［⏎］キーを押します。

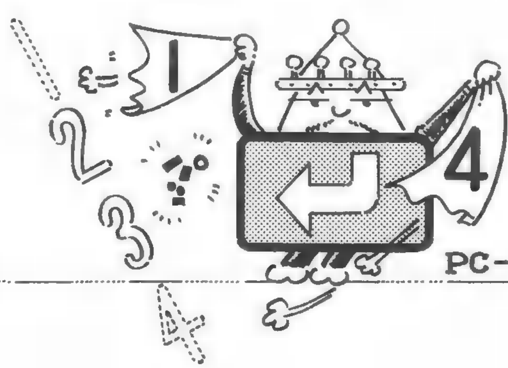
次のタイプをしてください。

【図35】

```
Run
PC-8201ﾊ
A4ｻｲｽﾞﾉ
ﾎﾟｰﾀﾌﾞﾙﾅ
ｺﾝﾋﾟｭｰﾀﾃﾞｽ
Ok
■
Load "  Save "  Files  List    Run
```

【図36】

```
List
1 PRINT "PC-8201ﾊ"
2 PRINT "A4ｻｲｽﾞﾉ"
3 PRINT "ﾎﾟｰﾀﾌﾞﾙﾅ"
4 PRINT "ｺﾝﾋﾟｭｰﾀﾃﾞｽ"
Ok
■
Load "  Save "  Files  List    Run
```



【図37】

```
1
4
List
2 PRINT "A4ｻｲｽﾞﾉ"
3 PRINT "ﾎﾟｰﾀﾌﾞﾙﾅ"
Ok
■
Load "   Save "   Files   List      Run
```

【図38】

```
new
Ok
List
Ok
■



Load "   Save "   Files   List      Run
```

1 ↵

4 ↵

画面から，その行が消えるわけではありませんが，メモリ内から行番号1と行番号4が消えます。LIST 命令で確認してみましょう(図37)。

RUN ↵ の場合も，もちろん残っている行番号2と行番号3しか実行されません。

メモリ内に記憶されているプログラム全体を消すには「NEW」(ニュー）命令を使います。

n e w ↵

NEW でプログラムを消した後は，LIST や RUN を実行しても何もせず，OK の文字だけが表示されます（図38)。

プログラムの作成・実行・修正をまとめると次のようになります。

①NEW 命令で記憶されているプログラムを消す。

②プログラムの命令を，行番号，命令，↵ の順にタイプする。これを繰り返す。↵ キーを押し忘れると，メモリ内には記憶されないので注意する。

③タイプは，行番号の順でなくてもよい。

④不要な行は，行番号のみタイプして ↵ キーを押すと，メモリ内から消える。

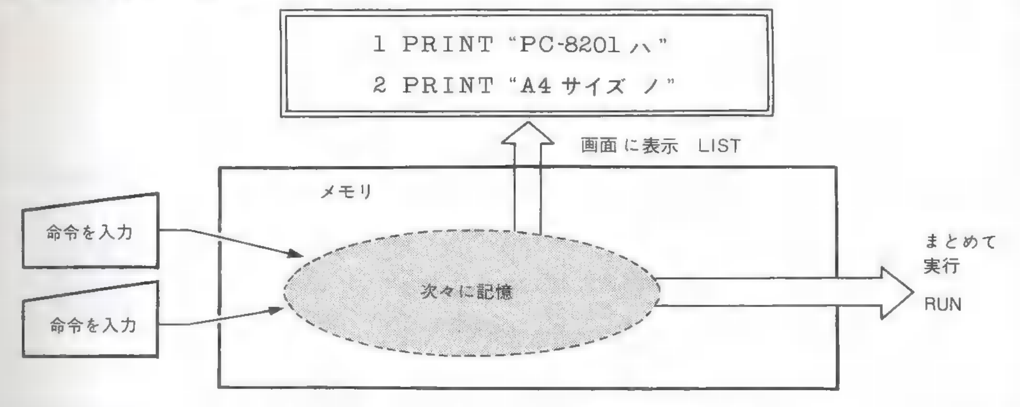
⑤命令は大文字でタイプしても，小文字でタイプしてもよいが，(“ ”）内はタイプされたままの文字が記憶される。

⑥記憶されているプログラムは，LIST 命令で画面に表示できる。

⑦記憶されているプログラムは，RUN 命令で何度でも実行できる。

BASIC の命令で扱うデータは，大きく「定数」と「変数」に分類され，定数と変数は，さらに文字型と数値型に分けられます(図40)。

【図39】 RUN, LISTの関連



このデータの種と型はきちんと区別する必要があります。なぜそのように分けなければならないのかは，実際にBASICでプログラムを作りはじめると，だんだんに理解できると思います。ここでは型について，一つずつ説明しましょう。

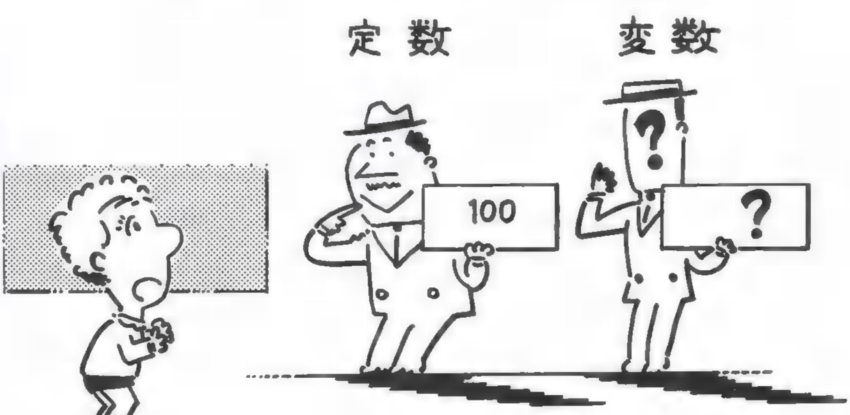
定数とは，プログラム内で固定的な値を持つデータのことで，変数とは，逆にプログラムが実行されるまでは値を持たない，または値が定まらないデータをいいます。「100」とか「パソコン」は定数であり，プログラム実行後，計算して求められる値や，キーボードから入力される値は変数です。

定数，変数のいずれでも，文字型データでは，画面の1文字や入力された1文字はメモリの中でも1文字（1バイト）として扱われますが，数値型データでは，メモリの節約と内部演算の効率化のために変換されたかたちで記憶されます。すなわち，画面にみる1桁の数字が，メモリ内では必ずしも1文字（1バイト）には該当しません。そのためBASICでは，計算に使う数値と住所などの文字およびその中に含まれる数字とは区別して扱わなければなりません。

文字定数は（"）で囲まれた文字の集まり（文字列といいます）で，255文字以内であれば，アルファベット，数字，カナ，記号が自由に組み合わせできます。

"PC-8201"

"BASIC"



【図40】 定数と変数の型

（例）

- 定数
  - 文字定数 ……… "PC-8201"　"ナマエハ?"
  - 数値定数
    - 整数型 ……… 123　−600　3%
    - 実数型
      - 単精度型 ……… 45.6　−925.7　25！
      - 倍精度型 ……… 1234567890　34.5#
- 変数
  - 文字変数 ……… A$　YM$
  - 数値変数
    - 整数型変数 ……… A%
    - 実数型変数
      - 単精度型変数 ……… B5　Y！
      - 倍精度型変数 ……… C9#

“プログラム”

整数型定数は−32768〜＋32767の範囲内の整数または%を使った整数で，メモリ内では2バイトに変換されて記憶されます。

256

−13800

39%

単精度型と倍精度型は，いずれも±38桁までの数値が扱えますが，その精度に差があります。

単精度型では，有効桁7桁で記憶され，6桁までしか有効数字は表示されず，それ以上はEを使った指数表示になります。メモリ内では4バイトに変換されて記憶されます。7桁以下の実数，Eを伴った指数，最後に！を使った数値はいずれも単精度として扱われます。

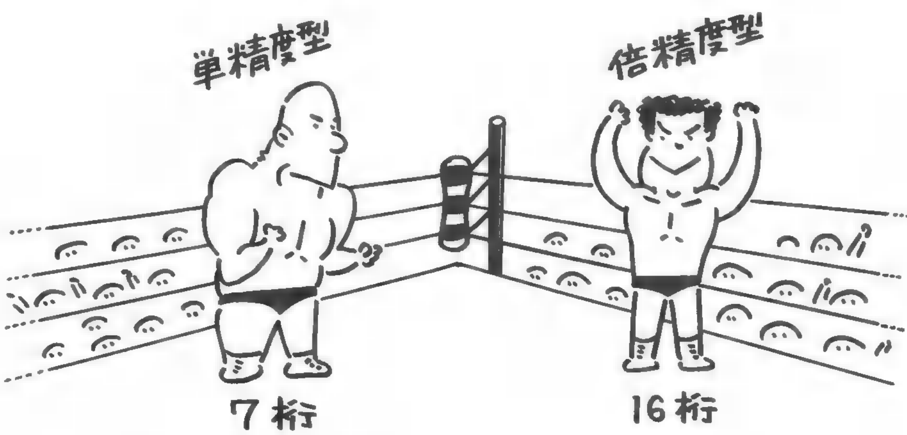
1.2345

1.2E-03

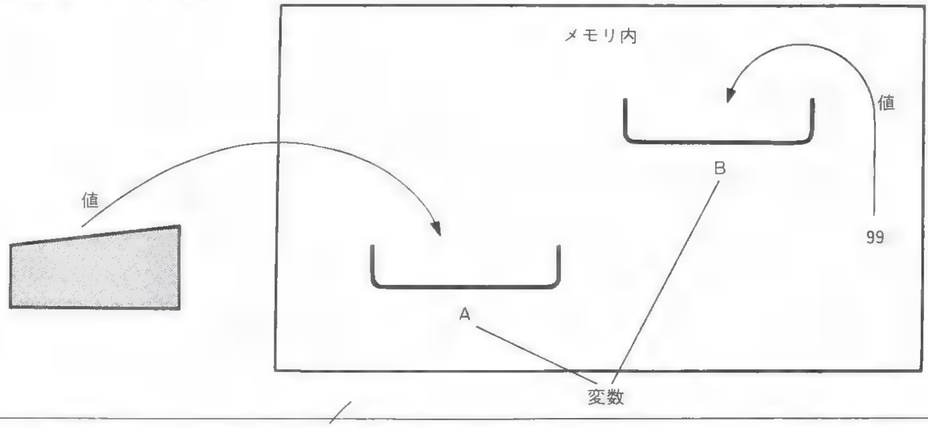
123！

倍精度型では，記憶および表示は有効桁16桁までで，それ以上はDを使った指数表示になります。8桁以上の数値，Dを使った指数，最後に#を使った数値は，いずれも倍精度として記憶されます。

1234567890



【図41】 メモリ内の記憶場所



1.234567D＋5

123＃

一方，変数の方は，プログラムが実行されるまで値をもたないので，その値を入れるためのメモリ内の記憶場所を指定します。変数には，他の変数と区別できるような名前をつけ，これを変数名といいます。変数名が指定されると，BASICは自動的にその場所をメモリ内に確保しますので，物理的なメモリ上の位置は知る必要がありません（図41)。

変数に一度値を入れると，別な値を入れるまで，何度使っても内容が変化しません。また，変数の内容がいろいろに変化しても，プログラムでは一つの変数名で扱うことができるので，定数に比較して命令を書くのが少なくてすみます。

変数名の規則

①1文字目はアルファベットとする。

②2文字目はアルファベットまたは数字とする。3文字目以降は，N_{82}-BASICでは無視される。

③変数名内のカナ文字と空白は無視される。

④変数名の中にBASICが使う文字（予約語という）を含んではならない。

⑤変数にも，文字型，整数型，単精度型，倍精度型がある。変数名の後に＄，％，！，＃をそれぞれつけて型を区別する。型が違えば変数名が同じでも別々の変数として扱われる。

A１＄ ｝型が違うので別の変数
A１＃ ｝同じ変数
A１２３４５＃

型宣言文字を省略すると，すべて単精度とみなされる。

プログラムのセーブとロード

PC-8201では，作成したプログラムは電源スイッチを切っても消えずに残っています。しかし，新しい別のプログラムを作成するために前のプログラムを消さなければならないことがでてきます。消さずに新しいプログラムをタイプすれば，前のプログラムへの追加の行とみなされて，２つのプログラムが混じってしまうからです。

PC-8201ではこのような時に，現在使っているプログラムを，メモリ内の空き領域に移して保存することができます。現在のプログラムが記憶されている場所も，保存用に移される場所も，どちらもPC-8201のメモリ(RAM）なのですが，区別しやすいように

前者を BASIC 領域，後者を RAM 領域と呼ぶことにします。

いいかえますと，PC-8201 では，これがこれまでのコンピュータの概念からいうと画期的なことなのですが，自分自身のメモリ内を，実行するためのプログラムの領域と外部記憶装置のような保存用領域の両方に使え，どちらも電源を切っても消えないのです。

RAM 領域は，フロッピーディスクのような外部記憶装置と同じように使えますが，メモリですから，外部記憶装置とは比べものにならないほど速く処理ができますし，メニュー画面上で保存されているファイルを操作できるという利点もあります。

現在，BASIC 領域にあるプログラムを RAM 上へ移すためには SAVE（セーブ）命令を使います。SAVE“ は [f·2] に登録されています。

サンプルのプログラムをタイプして，これを EX-2 の名前でセーブしてみましょう（図42)。

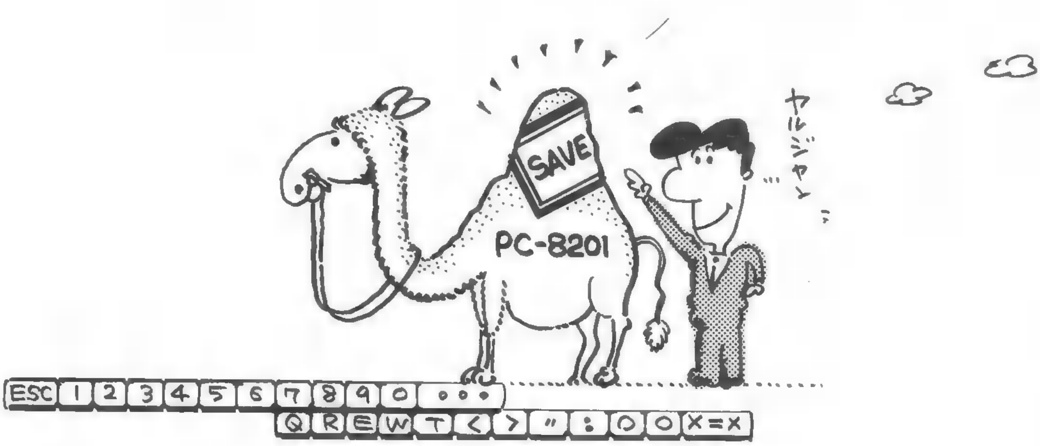
10 PRINT “サンプルプログラム” [⏎]
20 PRINT “セーブノレンシュウ” [⏎]
SAVE “EX-2” [⏎]

セーブが完了した後，LIST 命令を実行してみてください。プログラムがちゃんと表示されます（図43)。RAM へのセーブだけでなく，カセットテープなどへのセーブもそうですが，セーブの働きはプログラムを移すというより同じコピーを作るという方が正確です。したがって，セーブ後も，元のプログラムは変化せずに残っています。

【図42】

```
10 PRINT "ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ"
20 PRINT "ｾｰﾌﾞ ﾉ ﾚﾝｼｭｳ"
Save "EX-2"
Ok
■


Load "   Save "   Files   List     Run
```



【図43】

```
Save "EX-2"
Ok
List
10 PRINT "ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ"
20 PRINT "ｾｰﾌﾞ ﾉ ﾚﾝｼｭｳ"
Ok
■
Load "  Save "  Files   List     Run
```

ただし，PC-8201では，自分自身のメモリ内へのコピーという特別なセーブなため，少し違った処理をします。同じコピーを2つ持つのはメモリ節約の上からも好ましくないため，SAVE命令実行時は，名前の登録だけ行い，実際にはRAMへはすぐに書き込みません。NEW命令などにより，BASIC領域が消される時，自動的にRAMへ書き込みます。

ファンクションキーの10番を押し，メニュー画面へ戻してください。メニューへ戻す時も，その前にSAVE命令が実行されていると，RAMへの書き込みが行われます。同時にBASIC領域のプログラムは消されます。メニュー画面に戻すと，EX-2.BAがメニュー上に表示されているのを見ることができます（図44）。

EX-2は使用者がつけたプログラム名です。RAM上は，プログラムであってもデータであっても，どちらもファイルとして扱われるため，EX-2は正確にはファイル名と呼びます。ファイル名は，6文字以内のアルファベット，数字，カナ，一部の記号で自由につけることができますが，ほかのファイルと区別がつくように固有の名前にする必要があります。

EX-2の後に（.）で区切られてBAの文字があります。これは拡張子と呼ばれ，ファイルの種類を表しています。拡張子については，第1章のメニューの機能を参照してください。

では，もう一度BASICを選択します。1つプログラムをセーブした分だけメモリの空き領域を表す数字は減っています（図45）。

リストでプログラムが残っていないことを確認してください。SAVE命令だけではプログラムは消えませんが，RAM上に書かれた時点でBASIC領域から消されます。

【図44】

```
1983/05/01 12:27:51      (C) Microsoft #1
 BASIC      TEXT        TELCOM      EX-2.BA
 -.-        -.-         -.-         -.-
 -.-        -.-         -.-         -.-
 -.-        -.-         -.-         -.-
 -.-        -.-         -.-         -.-
 -.-        -.-         -.-         -.-
Load      Save      Name      List      28348
```

【図45】

```
NEC PC-8201 BASIC Ver 1.0 (C) Microsoft
28348 Bytes free
Ok
■




Load "  Save "  Files  List    Run
```

カセットテープのように，本体とは別な装置を使ってセーブする場合は，セーブした時点で2つの同一プログラムが別々の装置に存在します。その後，本体上のプログラム，つまりBASIC領域のプログラムに変更を加えても，カセットテープ上のプログラムには，もちろん影響がありません。別のいい方をすると，プログラムに少しでも修正があった場合は，そのつどセーブのやり直しが必要です。

PC-8201の特徴であるRAM上のプログラムでは，BASIC領域にプログラムがある時は，実際にはRAMへは移されていません。そのため，プログラムを修正した場合は，修正後のプログラムが自動的にRAMに移されます。

RAM上のプログラムをBASIC領域に呼び込む命令はLOAD（ロード）といいます。EX-2をもう一度BASIC領域にロードしてみましょう。LOAD"が[f·1]に登録されています。

LOAD "EX-2" [⏎]

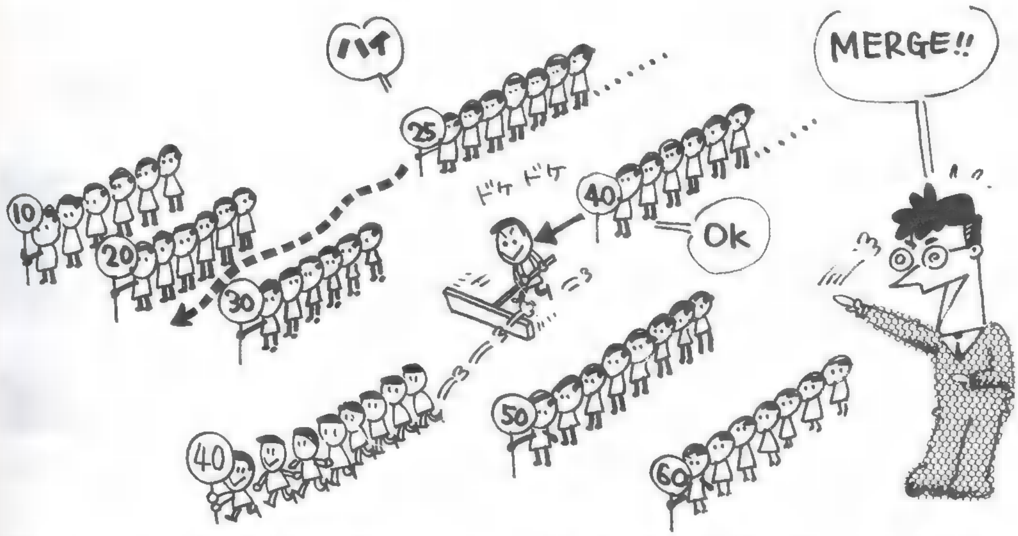
ロードが完了したら，LIST命令で確認してください（図46）。

カセットテープのような外部記憶装置からプログラムをロードする時は，何回ロードしてもカセットテープの内容は変わらずに残っていますが，RAM上のプログラムをBASIC領域に呼び込む時は，メモリの節約のため，RAM上のプログラムの方を消してしまいます。しかし，もう一度セーブしなくても，ニュー，ロードなどでメモリ上のプログラムが消される時やメニューへ戻る時は，自動的にBASIC領域のプログラムがRAM上に復元されます。

プログラムをセーブする時は，命令をコードに直した中間言語にします。中間言語の方が，命令をそのまま記録するよりも短くなるため，領域と時間の節約になるからです。拡

【図46】

```
Load "EX-2"
Ok
List
10 PRINT "ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ"
20 PRINT "ｾｰﾌﾞ ﾉ ﾚﾝｼｭｳ"
Ok
■
Load "  Save "  Files  List    Run
```



張子.BA が付いた BASIC プログラムとは，この中間言語のプログラムなのです。

しかし，コード化せずにそのままセーブしたい場合があります。RAM 上にセーブする時に，中間言語にコード化しないプログラムは，TEXT ファイルとしてセーブされますから，編集が楽です。

また，中間言語は，同じ PC シリーズのパソコンでもいろいろ異なるため，そのままでは他のパソコンへ移しても正しく機能しません。そこで中間言語にコード化しない元の BASIC 命令のままのプログラムでセーブしておく方法が必要になります。これをアスキーセーブといいます。アスキーセーブは次のように書きます。

SAVE "ファイル名", A

SAVE "CAS: ファイル名", A

アスキーセーブされたプログラムであっても，通常と同じようにロードできます。

BASIC 領域内にプログラムが入っている時，NEW をせずに新しいプログラムをタイプすれば，当然2つのプログラムは混じり合ってしまいます。ただし，ロードでは，NEW が自動的にかかるため，前のプログラムが消えてから新しいプログラムが入力されます。RAM またはカセットテープからプログラムをロードする時に，NEW をしないで，読み込む方法があります。この命令を MERGE といいます。

MERGE（マージ）命令

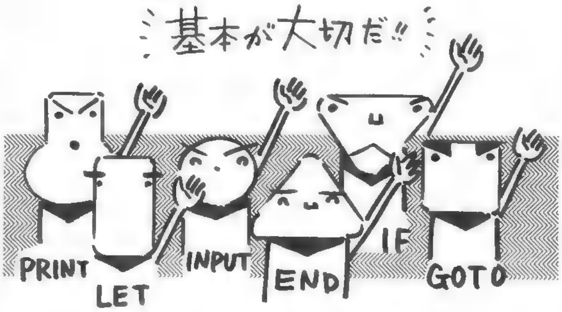
MERGE 命令を使うとNEWをかけずにプログラムをロードできます。その結果，読み込まれたプログラムは前のプログラムの追加修正分とみなされ，2つは1つのプログラムにまとまります。この MERGE を行う時に，プログラムはアスキーセーブしたもの，すなわち，.DO タイプでなければなりません。メモリ内のプログラムと MERGE のために読み込まれるプログラムの間に，同じ行番号があると，読み込まれる方が優先します。

② BASICの基本命令

N_{82}-BASIC には100以上の命令がありますが，自分でプログラムを作る場合には，まず基本の6つの命令をマスターしてください（表3）。

【表3】 6つの基本命令

PRINT	画面に表示する
LET	変数に値を入れる
INPUT	キーボードから変数に値を入れる
END	プログラムを終了させる
IF	条件判断する
GOTO	無条件にジャンプする



PRINT（プリント）命令

すでに，この命令は使ってきましたが，もう少し文法を詳しく説明しておきます。

PRINTは，画面に文字や数字を表示するための命令です。PRINTの後にダブルクォーテーション（“）で囲った文字を書くと，その文字が画面に表示されます。次の命令を入力して確認してみましょう（図47）。

PRINT “ABC” ⏎

PRINT “A$” ⏎

では，（“）なしでPRINTの後に文字を書いてみます（図48）。

PRINT　ABC ⏎

PRINT　A$ ⏎

このように，（“）で囲んだ場合は，指定の内容がそのまま表示されます。（“）で囲んであるので文字型定数として扱われています。（“）なしの場合は，AB（3文字目のCは無視されるため）およびA$という変

【図47】

```
print "ABC"
ABC
Ok
print "A$"
A$
Ok
■
Load "  Save "  Files  List    Run
```

【図48】

```
print ABC
 0
Ok
print A$

Ok
■
Load "  Save "  Files  List    Run
```

数の現在の記憶されている内容を表示します。AB は単精度型変数ですが，まだ使われていないので値は0（ゼロ）です。A$ は文字型変数ですが，同様にまだ使われていないので値は空です。

今度は PRINT の後に（“）で囲んだ数字を書いてみましょう（図49）。

PRINT “10−13”[⏎]

PRINT “+0.55”[⏎]

同様に（“）なしで PRINT の後に数字を書いてみます（図50）。

PRINT 10−13[⏎]

PRINT +0.55[⏎]

（“）で囲った場合は，どんな内容であっても，つまり数字であっても，BASIC 上は文字型データとみなされます。文字型データは，指定の内容がそのまま表示されます。文字型データは，実際には数字であっても計算には使うことができません。

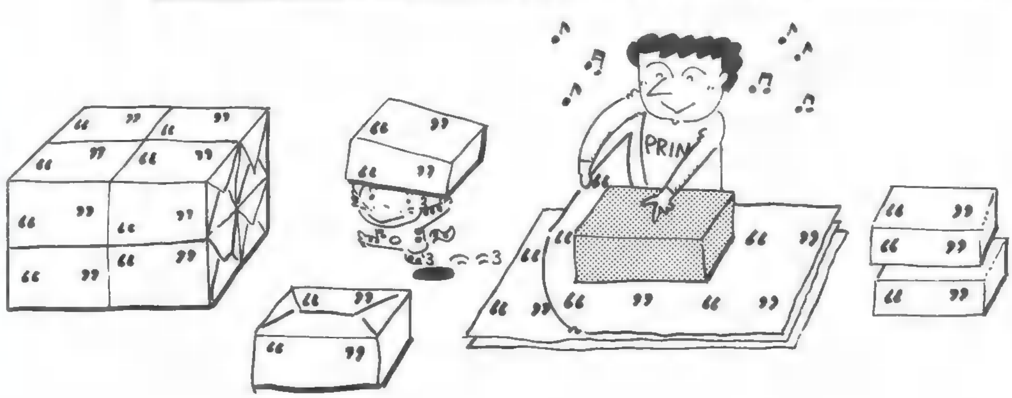
（“）なしの場合，つまり数値型データで

【図49】

```
print "10-13"
10-13
Ok
print "+0.55"
+0.55
Ok
■
Load "  Save "  Files  List    Run
```

【図50】

```
print 10-13
-3
Ok
print +0.55
 .55
Ok
■
Load "  Save "  Files  List    Run
```



【図51】

```
Run
A             B
AB
 123           456
 123  456
Ok
■
Load "   Save "   Files   List    Run
```

はその実質的な値を表示します。10−13では，計算結果の−3を表示します。+0.55では，小数点の前のゼロとプラスの符号を省略します。数値を表示する場合，1桁目はマイナス符号用に使います。したがって，プラスの時は，いつも1桁目が空白になります。このように，BASICでは，算用数字かどうかではなく，文字型として扱っているか，数値型で扱っているかで，処理が違ってきます。

PRINTの後に何も書かないと，単に改行が行われます。

PRINTの後には，逆に複数の文字列や数値を書くこともできます。区切りにはコンマ(,)，セミコロン(;)，空白が使われます。

(,)と(;)の機能の違いをみるため，次のプログラムを入力してみましょう（図51）。

10 PRINT "A", "B" ⏎
20 PRINT "A"; "B" ⏎
30 PRINT 123, 456 ⏎
40 PRINT 123; 456 ⏎

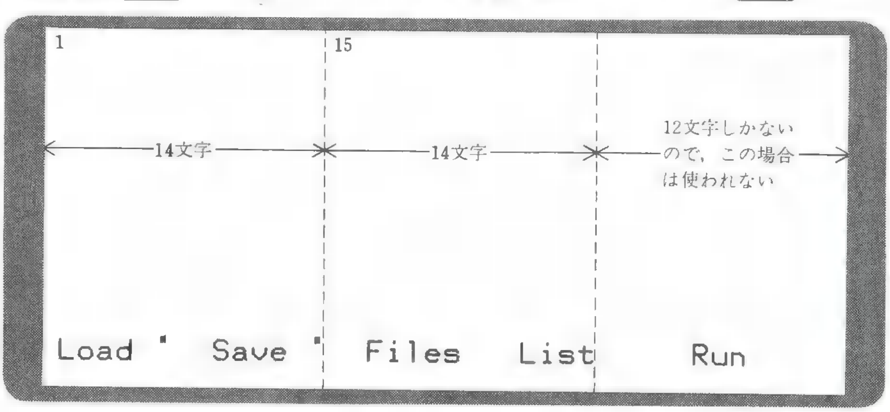
コンマ(,)で数値や文字列を区切った場合，画面上は14文字単位で表示されます。すなわち，最初は1文字目から，次は15文字目から，さらに続く時は改行して1文字目から，それぞれ表示されます（図52）。

セミコロン(;)で区切った場合，前の文字に続いて表示されます。ただし，数値の場合，直前にプラスマイナスの符号がつけられ，直後に1文字分空白があけられます。プラスの場合は符号がつけられないため，実際には2文字分の空白が区切りとしてあけられます。

文字型データでは空白を(;)の代わりに使うことができます。また，区切りの文字がなくてもかまいません。しかし，数値型データでは，空白は無視されるため区切りに使うことができません（図53）。

10 PRINT "A"; "B" ⏎
20 PRINT "C" "D" ⏎

【図52】



【図53】

```
Run
AB
CD
 1234
Ok
■

Load "  Save "  Files  List    Run
```

30 PRINT 12 34 ⏎

PRINT 命令が実行されると自動的に改行されますが，行の最後が（；）または（,）の時は続きがあるものとみなして改行されません。（；）と（,）の違いはデータの区切りの（；）と（,）の違いと同じです。

10 PRINT "A";"B"

↕ 同じ結果

10 PRINT "A";
20 PRINT "B"

LET（レット）命令

LET 命令は，変数に値を代入する働きをします。

LET A=123 ⏎

LET B$="N82BASIC" ⏎

LETの後に変数名，＝（イコール），代入すべき値の順に書きます。LET の文字は省略できるので，

A=123 ⏎

B$="N82BASIC" ⏎

と書いても同じです（図54)。

LET が実行されても，メモリ内の変数の変化は直接わからないので，PRINT で確認



【図54】

```
a=123
Ok
b$="N82BASIC"
Ok
■

Load "  Save "  Files  List    Run
```

【図55】

```
print a;b$
 123 N82BASIC
Ok
■

Load "  Save "  Files   List    Run
```

【図56】

```
a=123*29
Ok
print a
 3567
Ok
■

Load "  Save "  Files   List    Run
```

してみましょう（図55)。

代入すべき値は式であってもかまいません。式の値，つまり計算結果が代入されます（図56)。

アスタリスク（＊）のマークは，かけ算の記号です。

BASIC では，算術式を書くために，**表4**の記号を使います。

／記号を使った除算では，その数値で許される範囲まで計算して四捨五入します。小数点以下まで計算する必要のない場合は，￥記号で割算させます。MOD は，￥で割算した時の余りを求めるために使います。( ) は，何重にもすることができます。

次の計算を行ってみましょう。

PRINT　(123−5＊8)／2

PRINT　3∧3−6∧2

PRINT　10／3

PRINT　10#／3#

【表4】 算術式の記号

＋	加算
−	減算
＊（アスタリスク）	乗算
／（スラント）	除算
∧	指数演算
￥	整数の除算
MOD	剰余
(　)	演算順位を変えるためのカッコ

PRINT　10￥3

PRINT　10MOD3

答えはそれぞれ，41.5，−9，3.33333，3.333333333333333，3，1になります。

式の中に変数を含むこともできます。**図57**のプログラムを入力し，実行してみましょう。

変数名は，BASIC の文法に違反していない限り，プログラムごとに自由に決めてよい

【図57】

```
10 A=123
20 B=(A-5*9)/2
30 C=3^3-B
40 D%=C¥7
50 E%=C MOD 7
60 F#=E%/3#
70 PRINT A;B;C;D%;E%;F#
```

実行例

```
Run
 123  39 -12 -1 -5 -1.666666666666667
```

のですが，BASIC で役割を固定されている変数名がいくつかあり，これらについては使用者が勝手に使うことができません。このBASIC が役割を決めているものを予約変数といい，ある特別な値を記憶するために使います。予約変数の値をプログラム内で使うことはできますが，値を勝手に代入することはできません。

予約変数のうち，TIME＄ と DATE＄ には，LET により値を代入することができます。というよりも，この2つの予約変数には，LET でしか値を入れられません。

TIME＄ と DATE＄ は，＄のついた変数名から見当がつくように，文字型変数です。TIME＄ は現在の時間を，時：分：秒の形で，DATE＄ は，日付を，年／月／日の形で，それぞれ記憶していて，PC-8201 が動いている限り，時計およびカレンダーとして機能します。

プログラムなどと同様に，スイッチを切っても時計は止まりませんから，一度，正しい時間と日付をセット（代入）しておけば，PC-8201 は常に現在の時間と日付を教えてくれることになります。

例にならって，正しい時間と日付をセットしてください。各桁は必ず2桁です。

TIME＄＝"15: 05: 00" [⏎]

DATE＄＝"83／05／01" [⏎]



INPUT（インプット）命令

INPUT は，キーボードから変数へ，値を入れる働きをする命令です。

INPUT が実行されると，画面に？が1個表示されて止まり，キーボードからの入力を待ちます（図58）。

値を入力して［⏎］キーを押すと，入力された値が指定の変数へ入れられ（記憶され），先へ進みます。入力する値は変数の型に合わせます（図59）。

値を入力せず［⏎］キーのみ押した場合は，変数には値が入らず，INPUT 前の状態が保たれます（図60）。

INPUT 命令で指定する変数は，複数であってもかまいません。その時は，キーボードからデータを，コンマ（，）で区切って入力します。また，入力のデータは変数の型に

【図58】

```
input a
? ■




Load"   Save"   Files   List     Run
```

【図59】

```
input a
? 12345
Ok
print a
 12345
Ok
■
Load"   Save"   Files   List     Run
```

【図60】

```
input a
?
Ok
print a
 12345
Ok
■
Load"   Save"   Files   List     Run
```

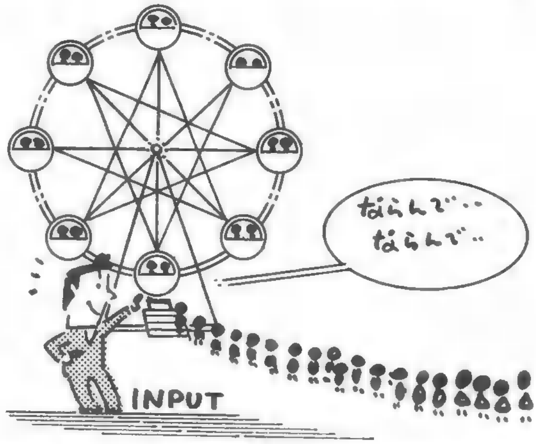
合わせなくてはなりません（図61）。

INPUT で指定した変数の数よりも入力したデータの数が多い時は，メッセージを表示して余分なものを無視します（図62）。

逆に変数の数よりも入力が少ない時は，2個のクエスチョンマーク（？？）を表示して，続きの入力を求めてきます（図63）。

残りの値を入力するまで INPUT命令は終わりません（図64）。

数値変数に数値以外を入力すると，エラー



【図61】

```
input a%,b,c#
? 3,681,6578000
Ok
■



Load"  Save"  Files  List    Run
```

【図62】

```
input a%,b,c#
? 1,234567,890,12
?Extra ignored
Ok
■


Load"  Save"  Files  List    Run
```

【図63】

```
input a%,b,c#
? 3,6500
?? ■




Load"  Save"  Files  List    Run
```

にはなりませんが，やり直しを求めてきます。REDO FROM START は，最初からやり直しのメッセージで（図65），正しいデータを入力すれば，先へ進みます。

INPUT 命令では，？の前に説明文を表示させることができます。これをプロンプト文といい，何を入力するのかをわかりやすくするために使います。プロンプト文は，命令に続けて（“）で囲んで書き，変数との区切りには，セミコロン（；）を書きます。

図66はプロンプト文付きの INPUT 命令の例です。

プロンプト文は，1つの INPUT 命令に1つしか書けません。図67の例は文法エラーになります。

これまで数値変数に値を入れる例ばかり説明してきました。今度は，文字変数にキーボードから値を入れてみましょう。

次のプログラムを入力してください。

10　INPUT　“ナマエハ”；N$ ⏎

【図64】

```
input a%,b,c#
? 3,6500
?? 44
Ok
■


Load"  Save"  Files  List    Run
```

【図65】

```
input a%
? abc
?Redo from start
? ■



Load"  Save"  Files  List    Run
```

【図66】

```
input "ｽｳﾘｮｳ";s
ｽｳﾘｮｳ? 24
Ok
input "A,B=";a,b
A,B=? 12,50
Ok
■
Load"  Save"  Files  List    Run
```

20　PRINT　N$ [⏎]

このプログラムを実行すると，画面には「ナマエハ?」のプロンプトが表れます。“saito”と入力します（図68)。

もう一度実行し，今度は（“）なしでsaitoと入力します（図69)。

LET命令で文字変数に文字定数を代入する時は，文字列を必ず（“）で囲む必要がありましたが，INPUT命令ではこのように，（“）で囲んでも囲まなくても文字データの入力ができます。通常は（“）なしで入力します。違いは次のとおりです。

（“）で囲った場合……文字列の中のコンマや空白がそのまま変数に代入される。ダブルクォーテーション（“）そのものは入力できない。

（“）で囲まない場合……コンマはデータの区切りとして扱われ，代入されない。文字列途中の空白と途中のダブルクォーテーション（“）は，データの一部とみなされるが，

【図67】

```
input "ｼﾝﾁｮｳ";s;"ﾀｲｼﾞｭｳ";t
ｼﾝﾁｮｳ? 170
?SN Error
Ok
■


Load "  Save "  Files  List    Run
```

【図68】

```
10 INPUT "ﾅﾏｴﾊ";N$
20 PRINT N$
Run
ﾅﾏｴﾊ? "saito"
saito
Ok
■
Load "  Save "  Files  List    Run
```

【図69】

```
Run
ﾅﾏｴﾊ? saito
saito
Ok
■


Load "  Save "  Files  List    Run
```

【図70】

```
Run
? " ｼﾌﾞﾔ , 2ﾁｮｳﾒ "
 ｼﾌﾞﾔ , 2ﾁｮｳﾒ
 14
Ok
■

Load "  Save "  Files  List   Run
```

【図71】

```
Run
?  ｼﾌﾞﾔ , 2ﾁｮｳﾒ
?Extra ignored
ｼﾌﾞﾔ           5
Ok
■

Load "  Save "  Files  List   Run
```

文字列前後の空白は無視される。

これを実際にためしてみましょう。次のプログラムを入力してください。

10 INPUT A$ ⏎

20 L=LEN (A$) ⏎

30 PRINT A$,L ⏎

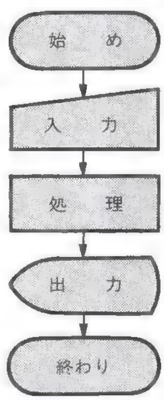
このプログラムを RUN 命令で実行させ,「 シブヤ , 2チョウメ 」を(“)付きで入力します(図70)。

もう一度実行し,同じものを(“)なしで入力します(図71)。

LEN は,文字列の長さをチェックする関数です。LEN の後の( )の中に,文字変数または(“)で囲んだ文字定数を指定すると,その実際の長さが値となります。

最初の例では空白やコンマも含めて14文字あります(濁点そのものも1文字と数えられます)。2番目の例では,コンマをデータの区切りとしてみなしたので,コンマ以降は,

【図72】 コンピュータ処理の基本形



IGNORED(無視)のメッセージを表示して変数には代入しませんでした。前の空白も無視されて文字数に数えられていませんが,コンマの直前の空白は,文字列の途中とみなされています。「シブヤ」とだけ入力したのであれば,4文字と数えます。

PRINT 命令,LET 命令に加えて INPUT 命令をマスターすると,少しプログラ

ムらしいプログラムが組めます。条件を入力し，処理し，結果を出力するというコンピュータ処理の基本形ができるからです(図72)。

図73を実行したのが図74です。

入力，処理，出力の基本パターンの応用例をあげておきましょう（図75，図76）。

【図73】

```
10 INPUT "suryo";S
20 T=50
30 K=S*T
40 PRINT "kingaku";K
```

【図74】

```
Run
suryo? 12
kingaku 600
Ok
■

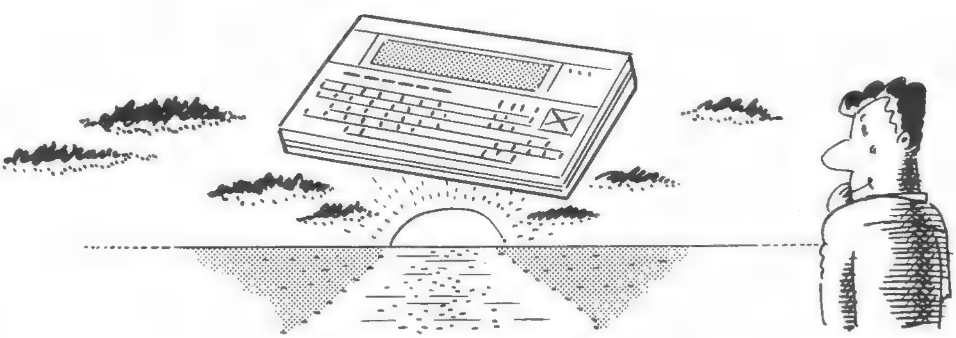

Load "  Save "  Files   List     Run
```

【図75】

```
10 INPUT "gankin";G
20 INPUT "riritu";R
30 INPUT "nensu";N
40 K=G*(1+R)^N
50 PRINT "ganrigokei";K
```

【図76】

```
10 INPUT "hankei";R
20 INPUT "ensyurritu";PAI
30 L=2*PAI*R
40 S=PAI*R^2
50 PRINT "ensyu";L
60 PRINT "menseki";S
```



END（エンド）命令

END は，プログラムを終了させる命令です。

すでに説明してきたプログラムでは END 命令がありませんが，プログラム自体は終了しています。BASIC においては，END 命令がなくても，順番に命令を実行し，これ以上実行すべき命令がない時には，自動的にプログラムを終了してコマンドレベルに戻るようになっています。

END 命令が必要なのは，任意の個所でプログラムを終了させたい時です。日常の例でいえば「全部の荷物を運んだら終わることにしてあるが，雨が降ってきたら途中でも終わる」などの場合です。したがって，1つのプログラムの中に，複数の END 命令が存在することもありうるわけです。

これ以降のプログラム例では，必ず END 命令を入れることにしましょう。

IF～THEN～ELSE命令

IF～THEN～ELSE 命令は，条件判断をする命令です。IF（イフ）の後に条件を書き，その条件どおりの時は THEN（ゼン）に続く命令を，条件と違う時は ELSE（エルス）に続く命令を実行します（図77)。

IF　A=1　THEN　PRINT　“OK”　ELSE　PRINT “NO”

ELSE 以降は省略できます。ELSE がない時に条件と違う時は，IF の命令は何も実行されず，次の命令に移ります（図78)。

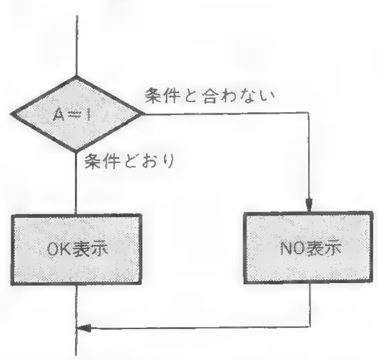
IF　A=0　THEN　C=C+1

条件は，BASIC では関係演算式または論理演算式で書きます。

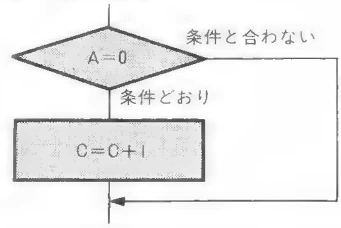
関係演算式は，2つの値を=，>，<の関係演算子で比較します。

等しい　　　　＝
等しくない　　＞＜　または＜＞
大きい　　　　＞
より大きい　　＞＝または＝＞

【図77】 条件判断の命令



【図78】 ELSEの省略



【図79】

```
10 INPUT "suryo";S
20 IF S>=10 THEN T=46 ELSE T=50
30 K=S*T
40 PRINT "kingaku";K
50 END
```

小さい	<
より小さい	<=または=<

図79のプログラム例を実行してみましょう。数量が10個以上の時は単価を46円，10個未満の時は単価を50円で金額を計算するものです（図80，図81）。

論理演算式は2つ以上の関係式の条件を調べたり，ビット操作やブール演算のために使

【図80】

```
Run
suryo? 9
kingaku 450
Ok
■

Load "  Save "  Files  List   Run
```

【図81】

```
Run
suryo? 10
kingaku 460
Ok
■

Load "  Save "  Files  List   Run
```

【図82】

```
10 INPUT "ﾃﾝｽｳ";T
20 IF T>=80 THEN PRINT "A"
30 IF T<80 AND T>=60 THEN PRINT "B"
40 IF T<60 THEN PRINT "C"
50 END
```

【図83】

```
Run
ﾃﾝｽｳ? 95
A
Ok
■

Load "  Save "  Files  List   Run
```

いますが，ここでは，NOT，AND，OR の論理演算子を使って2つ以上の関係式を同時に調べる例を示します。

否定　NOT

論理積（どちらも）　AND

論理和（どちらか）　OR

図82のプログラムを入力して，実行してみましょう（図83）。

GOTO（ゴーツー）命令

GOTO 命令は，行番号を指定してジャンプする命令です。GOTO の後に行番号を書くと，無条件にその行へジャンプします。GO TO とも書きます（図84）。

途中の行を飛ばして処理したり，元の行へ戻ったりするのに使います。

図85のプログラムを実行してみます（図86）。

ただし，この例は GOTO の働きを説明するためのもので，実際にプログラミングするときには，

10　PRINT　“A”;
20　PRINT　“C”;
30　END

ですませた方がもちろんよいのです。

GOTO 命令を IF 命令の中で使う時は，GOTO か THEN かのどちらかを省略することができます。

IF　A=0　THEN　GOTO　100　ELSE　GOTO　200

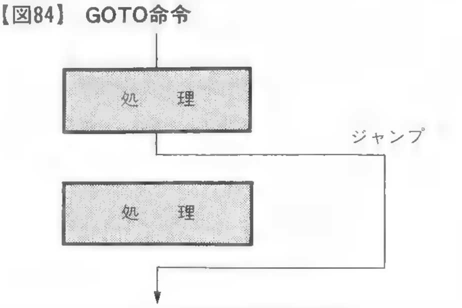
の代わりに

IF　A=0　THEN　100　ELSE　200

または，

IF　A=0　GOTO　100　ELSE　200

【図84】 GOTO命令



【図85】

```
10 PRINT "A";
20 GOTO 50
30 PRINT "B";
40 END
50 PRINT "C";
60 GOTO 40
```

【図86】

```
Run
AC
Ok
■

Load "  Save "  Files  List  Run
```

と書いてもかまいません。

図87のプログラムは図79を手直しして，何回でも繰り返せるようにしたものです。同じ命令を何回も繰り返すことをループ(LOOP)といい，データが複数ある場合に同じ命令を何度も書かずにすみます。繰り返すだけではプログラムは終わりませんから，ある値（ここでは−1）を入力したら終わることにしておきます。

終わりの印にする値は何でもよいのですが，実際のデータと同じでは区別がつきません。実際に−1のデータがありうる場合には，別な値を終わりのしるしにする必要があります(図88)。

−1を入力するまでは行番号10から行番号50の命令が何回でも実行されます。また同じ命令を繰り返すので変数には毎回違う値が次々と入れられ，処理の済んだ値は消されていきます。これが，データを全部，別々の変数に記憶させて消さないとすると，メモリはすぐ一杯になってしまうでしょう。繰り返しをすると，命令の数が少なくてすむだけでなく，変数の数も少なくてすむのです。

繰り返しを行う場合は，累計計算を行うことがあります。累計のために図87に次の命令を追加します（図89，図90）。

```
15 IF S=−1 THEN GOTO 60
45 G1=G1+S
46 G2=G2+K
47 C=C+1
60 PRINT "スウリョウケイ";G1
70 PRINT "キンガクケイ";G2
80 PRINT "ケンスウケイ";C
90 END
```

G1=G1+SやC=C+1は，数学的には誤りのような感じですが，LET 命令におけるイコール（=）は等しいの意味ではなく，右辺の結果を左辺に代入するための記号です。先に右辺を計算し，結果を左辺の変数に代入するので，たまたま同じ変数を使っていても計算がおかしくなることはありますん。G1=G1+S や C=C+1 は累計をとるときによく使われるパターンですので，覚えておくとよいでしょう。

【図87】

```
10 INPUT "suryo";S
15 IF S=-1 THEN END
20 IF S>=10 THEN T=46 ELSE T=50
30 K=S*T
40 PRINT "kingaku";K
50 GOTO 10
```

【図88】

```
suryo? 20
kingaku 920
suryo? 8
kingaku 400
suryo? -1
Ok
■
Load "  Save "  Files  List    Run
```

【図89】

```
10 INPUT "suryo";S
15 IF S=-1 THEN GOTO 60
20 IF S>=10 THEN T=46 ELSE T=50
30 K=S*T
40 PRINT "kingaku";K
45 G1=G1+S
46 G2=G2+K
47 C=C+1
50 GOTO 10
60 PRINT "ｽｳﾘｮｳｹｲ";G1
70 PRINT "ｷﾝｶﾞｸｹｲ";G2
80 PRINT "ｹﾝｽｳ ｹｲ";C
90 END
```

【図90】

```
kingaku 506
suryo? -1
ｽｳﾘｮｳｹｲ 47
ｷﾝｶﾞｸｹｲ 2186
ｹﾝｽｳ ｹｲ 4
Ok
■
Load "  Save "  Files  List  Run
```

③ 入出力のプログラム

データの出力

コンピュータで処理された結果を取り出すことを出力（またはアウトプット）といいます。パソコンにおいて，もっとも普通の出力方法は画面に表示することです。BASIC では，PRINT 命令がその働きをしています。

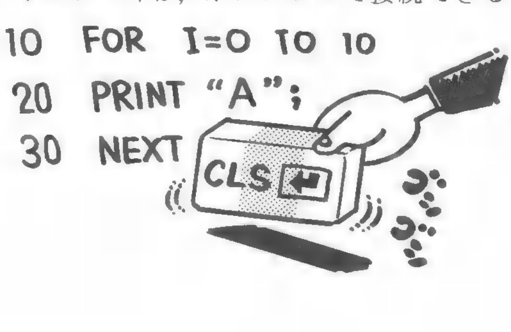
画面に出力するのは PRINT 命令だけで可能ですが，より見やすい形式にするためのいくつかの命令があります。

CLS 命令

画面に表示されているものをすべて消します。[C][L][S][↵]と入力してみてください。ただし，画面の一番下の行のファンクションキーの内容の表示は消えません。ファンクションキーの内容表示を消すのは SCREEN 命令です。

SCREEN（スクリーン）命令

使用する画面が，PC―8201 自身の液晶ディスプレイか，オプションで接続できる



CRT ディスプレイかを指定する命令です。同時にファンクションキーの内容表示をするかしないかも指定できます。オプションのCRTは接続されていないものとして，次の命令を交互に入力してみてください。

SCREEN　0，0

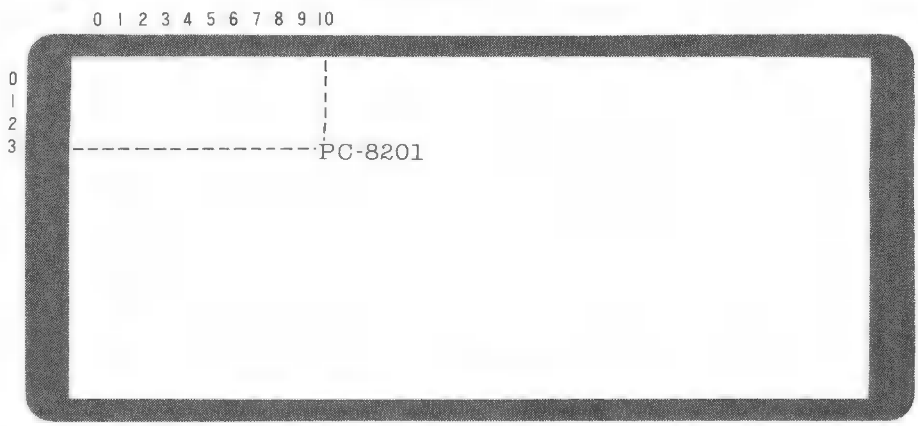
SCREEN　0，1

2つ目の数字が0の時，ファンクションキーの内容表示をせず，1の時表示をします。前の0は，PC-8201自身の液晶ディスプレ

【図91】

```
10 SCREEN 0,0:CLS
20 LOCATE 10,3:PRINT "PC-8201"
```

【図92】



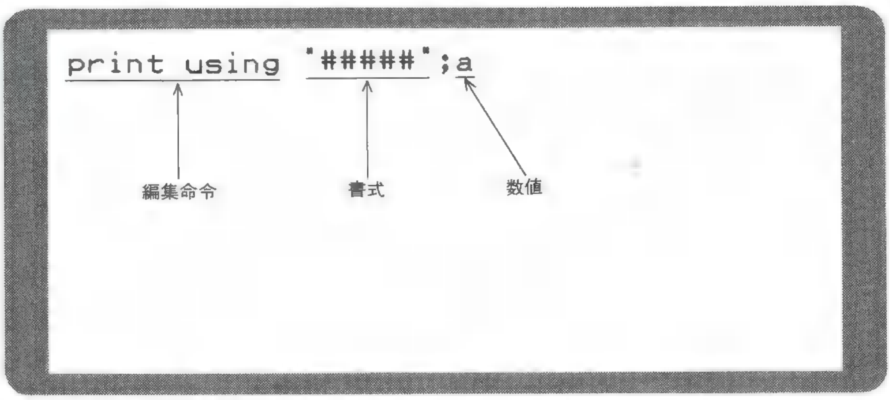
【図93】

```
10 SCREEN 0,0:CLS
20 LOCATE 10,0:PRINT "suryo"
30 LOCATE 20,0:PRINT "tanka"
40 LOCATE 30,0:PRINT "Kingaku"
50 Y=1
60 LOCATE 10,Y:INPUT S
70 IF S=-1 THEN GOTO 170
80 IF S>=10 THEN T=46 ELSE T=50
90 LOCATE 20,Y:PRINT T
100 K=S*T
110 LOCATE 30,Y:PRINT K
120 G1=G1+S
130 G2=G2+K
140 C=C+1
150 Y=Y+1:IF Y=7 THEN 10
160 GOTO 60
170 LOCATE 2,Y:PRINT "ｺﾞｳｹｲ"
180 LOCATE 10,Y:PRINT G1;"ｺ"
190 LOCATE 20,Y:PRINT C;"ｹﾝ"
200 LOCATE 30,Y:PRINT G2;"ｴﾝ"
210 END
```

【図94】

```
            suryo     tanka     kingaku
            ? 10       46        460
            ? 6        50        300
            ? 70       46        3220
 ｺﾞｳｹｲ        86 ｺ      3 ｹﾝ      3980 ｴﾝ
Ok
■
```

【図95】



イを意味し，CRT ディスプレイを接続する時は，ここを1にします。

LOCATE（ロケート）命令

カーソルを任意の位置に移動します。LOCATE の後に，0～39の水平座標，0～7の垂直座標を指定します。画面の範囲の横39，縦7を越えて指定しても，39，または7とみなしますが，255以上の値を指定するとエラーになります。

C L S で画面を消した後，図91の命令を入力してください。

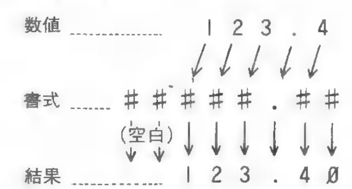
任意の位置にカーソルが移動できると，画面表示がみやすくなります（図92）。

図93の例を図79と比較してみるとよくわかります（図94は実行例）。

PRINT USING（プリントユージング）命令

出力する数値をみやすい形に編集して出力する命令です。PRINT USING に続けて，

【図96】 プリントユージング命令



どのように編集するかの指定（これを書式制御と呼びます）を（“）で囲んで書き，（;）で区切って編集したい数値を書きます。数値は定数，変数，式のいずれでもかまいません（図95）。

PRINT USING 命令の中では，#や＊に特別な意味を持たせています。これらの文字を書式制御文字と呼びます。書式制御は，書式制御文字をいくつか組み合せて指定します。

数値の編集では，必ず書式の右側から（右詰め），すなわち1の位から数字を代入します。小数点がある場合は，小数点を基準にします。書式指定よりも有効桁が少なく，書式

の左側が余った時は，空白等で埋められ，数字の頭に0（ゼロ）が出力されることはありません（図96)。

書式制御文字を表5に掲げておきます。

図97を実行して，それぞれの制御文字の機能を理解してください（図98は実行例です)。

書式制御文字以外の文字を書式の（“）の中に含めると，その文字がそのまま出力されます。これは書式制御文字であっても，＊を1つだけ書いた時のように，書式の文法と違う書き方をした場合は同じです。例えば，1234を“＊######エン”の書式で編集すると，＊　　1234エンと出力されます。

【表5】書式制御

制御文字	意味	働き
#	数字の桁数	必要な桁数分の#を書く。先頭の#が余った時は空白を出力する
.	小数点	小数点を出力する
,	3桁毎のコンマ	#の中に(,)を書くと3桁ごとに(,)を出力する。(,)は#の中のどこにあってもよいが，#の先頭と小数点の後では働かない
+	±の出力	書式の最初または最後に書くと，符号を出力する
−	−の出力	書式の最後に書くと，数値の値が負の時だけ符号を出力する
**	*で埋める	書式の最初に*を2つ書くと，余った位置に*を出力する
¥¥	¥出力	書式の最初に¥を2つ書くと，有効数字の直前に¥を出力する
**¥	——	**と¥¥の機能
^^^^	指数表示	#の後に書くと，指数形式で出力する

【図97】

```
10 A#=123.456789#:B#=-987.654321#
20 C#=12345678#
30 PRINT USING"#####.##";A#
40 PRINT USING"#########.##";B#,C#
50 PRINT USING"+#####.##";A#
60 PRINT USING"#####.##+";A#
70 PRINT USING"#####.##-";B#
80 PRINT USING"**####.##-";B#
90 PRINT USING"¥¥####.##-";B#
100 PRINT USING"**¥###.##-";A#
110 PRINT USING"#########,";C#
120 PRINT USING"######^^^^";C#
130 END
```

【図98】

```
   123.46
      -987.65 12345678.00
   +123.46
   123.46+
   987.65--
 ***987.65-
   ¥987.65--
 **¥123.46
 12,345,678
  12346D+03
```

1つの書式に続けて複数の数値を書くことができます。また，1組の（“）の中に，複数の書式指定を書くこともできます。書式と数値の割り当ては図99のように行われます。

【図99】 書式と数値の割り当て

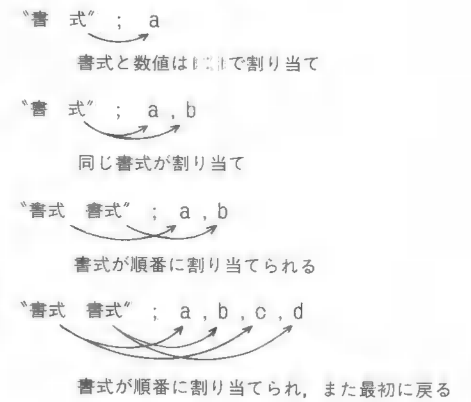


図93を手直しして，各項目の桁ぞろえをしたのが，次の図100です（図101はその実行例)。

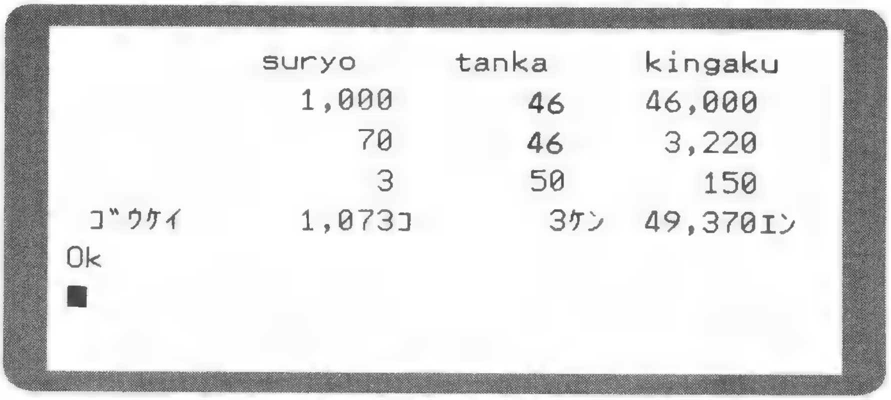
PSET（ピーセット）命令と PRESET（ピーリセット）命令

PSET 命令は，画面上の指定した座標にドットをセットする，つまり点を表示し，

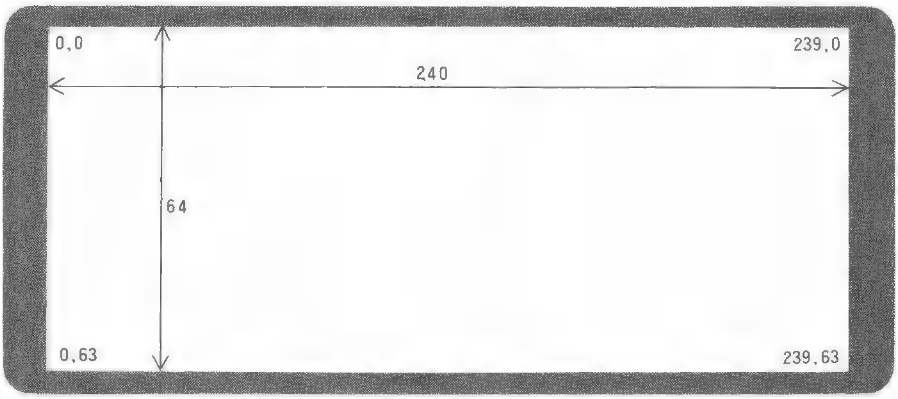
【図100】

```
10 SCREEN 0,0:CLS
20 LOCATE 10,0:PRINT "suryo"
30 LOCATE 20,0:PRINT "tanka"
40 LOCATE 30,0:PRINT "kingaku"
50 Y=1
60 LOCATE 10,Y:INPUT S
65 LOCATE 10,Y:PRINT USING "#####,";S
70 IF S=-1 THEN GOTO 170
80 IF S>=10 THEN T=46 ELSE T=50
90 LOCATE 20,Y:PRINT USING"#####,";T
100 K=S*T
110 LOCATE 30,Y:PRINT USING "#####,";K
120 G1=G1+S
130 G2=G2+K
140 C=C+1
150 Y=Y+1:IF Y=7 THEN 10
160 GOTO 60
170 LOCATE 2,Y:PRINT "ゴウケイ"
180 LOCATE 10,Y:PRINT USING"#####,コ";G1
190 LOCATE 20,Y:PRINT USING"#####,ケン";C
200 LOCATE 30,Y:PRINT USING"#####,エン";G2
210 END
```

【図101】



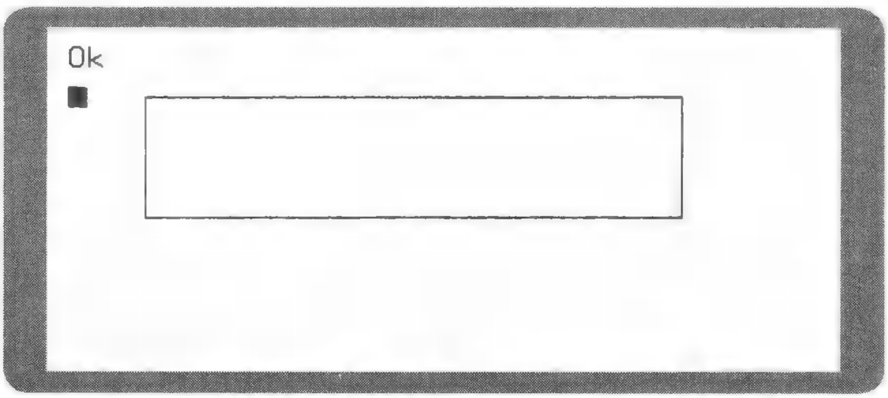
【図102】



【図103】

```
10 SCREEN 0,0:CLS:J=5
20 FOR I=30 TO 200
30 PSET(I,J,1)
40 NEXT
50 IF SW=0 THEN SW=1:J=40:GOTO 20
60 SW=0:I=30
70 FOR J=5 TO 40
80 PSET(I,J,1)
90 NEXT
100 IF SW=0 THEN SW=1:I=200:GOTO 70
```

【図104】



PRESET 命令は，指定した座標のドットをリセット，つまり点を消します。

PC-8201 の画面は，文字（キャラクタ）単位では横 0 —39，縦 0 — 7 ですが，ドット単位では横 0 —239，縦 0 —63の座標を持ちます（図102）。

PSET 命令，PRESET 命令はともに，その後に，水平座標，垂直座標，ファンクションコードをコンマで区切って書きます。ファンクションコードには0～255の数を指定します。ファンクションコードが奇数の場合と省略された場合は，本来の機能どおりに働きます。偶数の場合には，PSET はPRESET として，PRESET は PSET として働きます（図103はプログラム例，図104はその実行例です）。

データの入力

コンピュータに処理の対象となるデータを与えることを入力（またはインプット）といいます。パソコンにおいてもっとも普通の入力方法はキーボードのタイプです。BASIC では INPUT 命令がその働きをします。

しかし，それ以外の命令で入力することもできます。INPUT 命令以外の入力命令を説明しましょう。

LINE INPUT （ラインインプット)命令

キーボードから指定の文字変数に値を入力するための命令です。最大255文字まで入力可能で，［⏎］キーを押すまでにタイプされたどんな文字でも受け入れます。INPUT 命令と似た命令ですが，どんな文字でも入力できることのほか命令が実行された時，画面に？が表示されないこと，指定できる変数は1個の文字変数だけであることが違います。

LINE INPUT 命令は(，)や(“)を入力

【図105】

```
10 SCREEN 0,0:CLS
20 LOCATE 5,0:PRINT "ﾅﾏｴ"
30 LOCATE 20,0:PRINT "ﾃﾞﾝﾜ"
40 Y=1
50 LOCATE 5,Y:LINE INPUT A$
60 IF A$="end" THEN END
70 LOCATE 20,Y:LINE INPUT B$
80 Y=Y+1
90 IF Y>6 THEN GOTO 10
100 GOTO 50
```

【図106】

```
     ﾅﾏｴ            ﾃﾞﾝﾜ
     saito          499-2671
     yamada         862-3573
     kojima         743-4305
     end
Ok
■
```

したい時はもちろんですが，画面に？を出したくない理由で使う例もたくさんあります。図105にその例を示します（図106はその実行例です）。

INPUT＄（インプットダラー）命令

指定された長さの文字の入力を行うもので，正確には命令ではなく関数です。

INPUT＄関数は，指定された文字数の入力がされるまで待ち続け，［⏎］キーが押されても，１文字分の入力として扱います。指定の長さ分の入力が終わると先へ進みます。入力された文字は画面表示されません。

INPUT＄関数はキーボードからの入力のほかに，ファイルからの入力にも使われます。ファイルからの入力についてはファイルの項で説明しますので，ここではキーボードからの入力例だけ示します（図107はプログラム，図108は実行例です）。

INPUT＄は関数なので単独では使えません。LET 命令を使って，変数に代入してから使っています。また INPUT＄では，プロンプト文は使えませんし，入力した内容の表示もしないので，別に PRINT 命令を使っています。そうまでして INPUT＄を使うメリットは，必ず決まった長さ分入力させたい時に，長すぎる，短かすぎるということが起こらないからです。

INKEY＄（インキーダラー）命令

どのキーが押されたかを調べる関数です。INKEY＄の内容を変数に代入するか，直接 IF 命令を使ってチェックします。どのキーも押されていない時は，INKEY＄の値はヌルストリング，つまり空です。

INKEY＄の活用方法はいろいろあります

【図107】

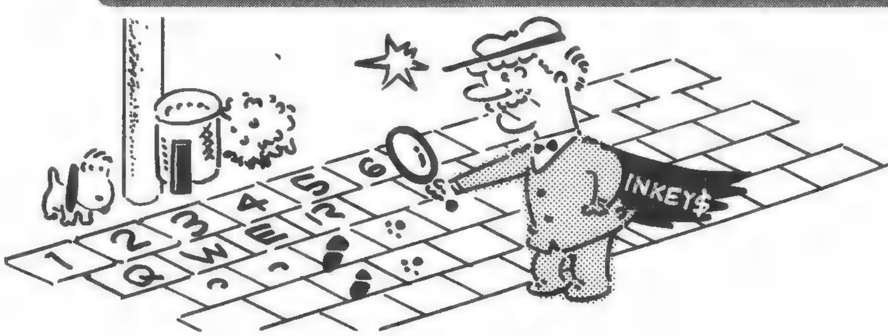
```
10 PRINT "ハ゛ンコ゛ウ ヲ 2ケタ テ゛ ニュウリョク"
20 A$=INPUT$(2)
30 PRINT A$
```

【図108】

```
Run
ハ゛ンコ゛ウ ヲ 2ケタ テ゛ ニュウリョク
12
Ok
■



Load"  Save"  Files  List   Run
```



が，代表的な例を2つあげておきましょう。

最初の例は，あるキー（ここでは[ESC]キー）が押されたかどうかをチェックし，押された時だけ特別な処理をします（図109）。CHR$（27）は[ESC]キーのことです。[ESC]キーがキャラクタコード表の27番目にあることを確認してください。

この方法を応用すると，ゲームで砲台を移動したり，スペースバーを押した時，射撃することも可能です。また，表形式のプログラムで自由にカーソルを移動し，任意の欄から入力することもできます。

2番目の例は，画面に情報を表示し，見終わったら先へ進むという時に使うものです。ここではINKEY$は画面を任意の時間止める働きをしています。もし，INKEY$でチェックして，何かキーが押されるまで先へ進めないという方法をとらない場合，画面の表示はパッパッと変化して，とうてい読むひまはありません（図110）。

READ（リード）命令

これまで説明してきた命令と関数は，キーボードからの入力でしたが，READ命令はプログラム内で定義済みの値を変数に代入する働きをします。値は，あらかじめ定数としてプログラム内に組み込まれていますから，入力というのは少しおかしい気もしますが，このREAD命令でしか，定義済みの値を使うことができないので，コンピュータへの入力命令と考えます。

READ命令は，DATA（データ）命令で定義されている値を1つずつ取り出し変数に代入します。READ命令に続く変数の数が複数であれば，DATA文から同じ数だけ値が取り出されて代入されます。定義されている値と，変数の型は一致していなければなりません。

一度取り出された値は，RESTORE（リ

【図109】
```
10 PRINT "ESCｷｰ ｦ ｵｽﾄ ﾚｲｶﾞｲ ｼｮﾘ"
20 A$=INKEY$
30 IF A$="" THEN GOTO 20
40 IF A$=CHR$(27) THEN GOTO 70
50 PRINT "MAIN ROUTINE"
60 GOTO 20
70 PRINT "ﾚｲｶﾞｲ ROUTINE"
80 GOTO 20
```

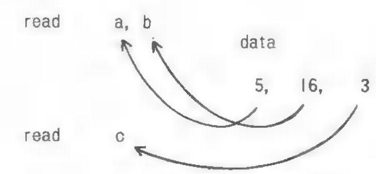
【図110】
```
10 CLS
20 LOCATE 10,0:PRINT "* INKEY$ ﾉ ｾﾂﾒｲ *"
30 LOCATE 10,2:PRINT "ﾄﾞﾉ ｷｰ ｶﾞ ｵｻﾚﾀﾉｶ"
40 LOCATE 10,3:PRINT "ﾁｪｯｸ ｽﾙ ｶﾝｽｳ "
50 LOCATE 10,4:PRINT "ﾖﾐｵﾜｯﾀﾗ ﾅﾆｶ ｷｰ ｦ "
60 LOCATE 10,5:PRINT "ｵｼﾅｻｲ"
70 A$=INKEY$
80 IF A$="" THEN GOTO 70
90 CLS
100 LOCATE 10,2:PRINT "ﾄﾞﾉ ｷｰ ﾓ ｵｼﾃ ｲﾅｲ ﾄｷﾊ"
110 LOCATE 10,3:PRINT "ﾇﾙｽﾄﾘﾝｸﾞ ｶﾞ ｱﾀｲ ﾄ ﾅﾙ"
```

ストア）命令がない限り，二度と使われませんので，READ 命令が実行されるごとに DATA 命令内の次の値へ移っていきます（図111)。

READ 命令で取り出して使われない値があっても，つまり DATA 命令で定義した値の方が多くてもかまいませんが，逆に READ 命令で値を取り出して変数に代入しようとした時，取り出すべき値が残っていない場合はエラーになります。

関連する DATA 命令を説明しておきます。DATA 命令は READ 命令で取り出される値を定義するもので，プログラムのどこにあってもかまいません。DATA の後に，値をコンマ（，）で区切って書いていきます。READ 命令の変数の型と一致していれば，どの型でも定義でき，文字もコンマや前後の空白を含まなければ，（“）なしで書けます。定義しておきたい値が1つの DATA 命令内に書き切れない時は，何行の DATA 命令に分けてもかまいません。READ 命令は行番号の小さい DATA 命令から扱っていきます。

【図111】 READ命令



10 DATA 5, イトウ, 7, ウエダ, 8, オノ, 10, カワシマ
20 DATA 11, キムラ

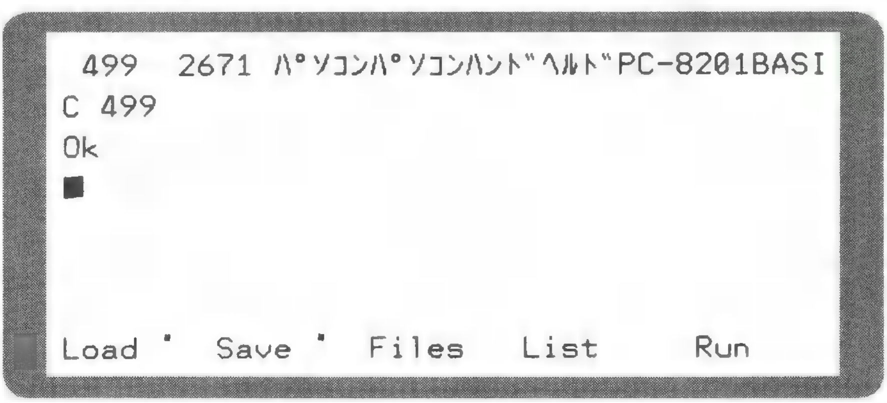
RESTORE（リストア）命令

RESTORE 命令を使うと，READ 命令で取り出す値の順番を変えることができます。

【図112】

```
EX-22
10 CLS
20 DATA 499,2671
30 DATA ﾊﾟｿｺﾝ,ﾊﾝﾄﾞﾍﾙﾄﾞ,PC-8201
40 DATA BASIC,TEXT,TELCOM
50 READ A,B
60 READ C$
70 RESTORE 30
80 READ D$,E$,F$,G$
90 RESTORE
100 READ H
110 PRINT A;B;C$;D$;E$;F$;G$;H
120 END
```

【図113】



RESTORE 命令で，特定の DATA 命令の行を指定すると，READ 命令で次に値を取り出す時は，指定の DATA 命令から開始します。

行番号なしの RESTORE 命令が実行されると，READ 命令は，一番小さな行番号をもつ DATA 命令，つまり DATA の最初から値を取り出します。

図112のプログラム例を実行してみます（図113は実行例）。

READ 命令により DATA の最初から，データが取り出されて変数に入れられますが，RESTORE 命令により，取り出す DATA の順序が変わっています。

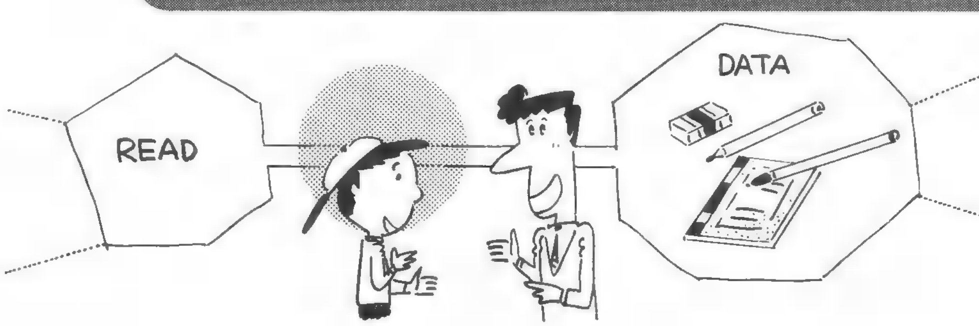
図114のプログラムは，コードを入力し，そのコードによりあらかじめ定義済みの商品名と単価を表示するものです（図115は実行例）。

【図114】

```
10 INPUT "ｺｰﾄﾞ";C
20 IF C=999 THEN END
30 RESTORE
40 READ A,N$,T
50 IF C<>A THEN GOTO 90
60 PRINT "ｼｮｳﾋﾝﾒｲ ";N$
70 PRINT "ﾀﾝｶ";T
80 GOTO 10
90 IF A<>999 THEN GOTO 40
100 BEEP:PRINT "ｶﾞｲﾄｳ ﾅｼ !!"
110 GOTO 10
120 DATA 11,ｴﾝﾋﾟﾂ,50,12,ﾎﾞｰﾙﾍﾟﾝ,100
130 DATA 15,ﾉｰﾄ,250,16,ｹｼｺﾞﾑ,40
140 DATA 21,ﾊﾞｲﾝﾀﾞｰ,600,999,end,999
```

【図115】

```
ｺｰﾄﾞ? 16
ｼｮｳﾋﾝﾒｲ ｹｼｺﾞﾑ
ﾀﾝｶ 40
ｺｰﾄﾞ? 12
ｼｮｳﾋﾝﾒｲ ﾎﾞｰﾙﾍﾟﾝ
ﾀﾝｶ 100
ｺｰﾄﾞ? ■
Load "  Save "  Files  List    Run
```



4 繰り返しのプログラム

FOR～NEXT

コンピュータ処理を行う時は，たいてい多くのデータを扱います。各データの1件ごとに命令を書いていてはプログラムは膨大なものになってしまいます。しかし，各データの内容は違っていても，計算や表示の手順，つまり命令は同じものなので，命令を最初から繰り返すようにすれば，1回書くだけで済みます。GOTOのところでも説明しましたが，ループ（繰り返し）は大量のデータ処理をする時の基本的な方法です（図116)。

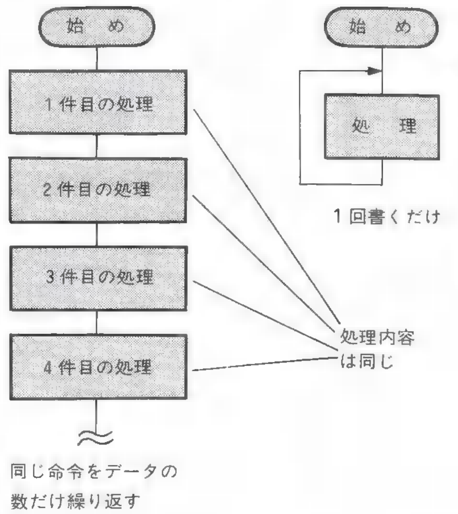
繰り返しをするプログラムの作成には2つの方法があります。1つは，GOTO命令で最初に戻して，何度でも同じ命令をグルグル回るようにするものです。この時はプログラムが永久に終わらなくなってしまうので，終わるための条件を決めておいて，IFで判断してENDにつながるようにします。

IFで終わりを判断するときも，2つの方法があります。ある特別な値が入力された時に終わるようにしておくのと，繰り返しの回数を決めておいてその回数に達したら終わるものとです。

図117はENDが入力されたら終わるもの（図118はその実行例)，図119は4回繰り返したら終わるもの（図120はその実行例）のそれぞれの例です。両方のプログラムで使われているTAB（タブ）は，PRINT命令の中でだけ使われる関数です。TABは，TABに続く( )内の数字の位置までカーソルを移動します。詳しくは，関数の項でまた説明します。

処理されるデータの件数は固定的でないのが普通です。したがって，あらかじめデータの件数を条件にして繰り返すのはよい方法ではありませんが，ある部分だけを考えると回数が決まっていて繰り返すことがよくあります。画面に6行ずつ表示するとか，5回ごとに平均をとるとかの場合です。

【図116】 大量データ処理の基本形



【図117】

```
10 SCREEN 0,0:CLS
20 LOCATE 0,6:INPUT "ｵﾜﾘﾅﾗ end";IN$
30 IF IN$="end" THEN END
40 I=I+1
50 LOCATE 0,I:PRINT TAB(I) "ｸﾘｶｴｼ"
60 GOTO 20
```

【図118】

```
クリカエシ
 クリカエシ
  クリカエシ
   クリカエシ

オワリナラ end? ■
```

【図119】

```
10 SCREEN 0,0:CLS
20 I=I+1:IF I>4 THEN END
30 PRINT TAB(I) "クリカエシ"
40 GOTO 20
```

【図120】

```
クリカエシ
 クリカエシ
  クリカエシ
   クリカエシ
Ok
■
```

繰り返し回数があらかじめ決っていて同じ命令をループさせるためには，IF と GOTO を使う方法のほかに専用の FOR～NEXT という命令があります。

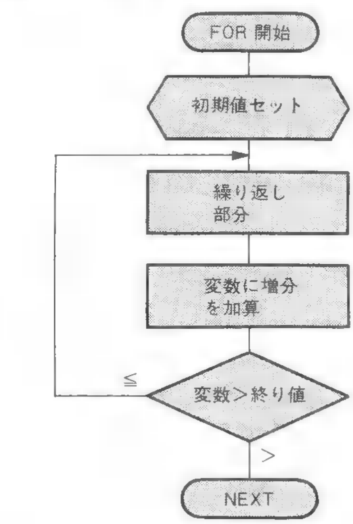
FOR（フォア）～NEXT（ネクスト）命令

FOR と NEXT にはさまれたいくつかの命令を指定回繰り返す働きをします。FOR の後に変数名，初期値，終り値，増分を次の書式で書きます。

FOR I=1 TO 10 STEP 2

変数 I に初期値 1 が代入されて，1 回目のループが実行されます。それから増分 2 が変数に加算され，終り値を超えていないかチェ

【図121】 FOR NEXTの基本形



【図122】

```
10 SCREEN 0,0:CLS
20 FOR I= 1 TO 4
30 PRINT TAB(I) I;"ｶｲﾒ ｸﾘｶｴｼ"
40 NEXT
```

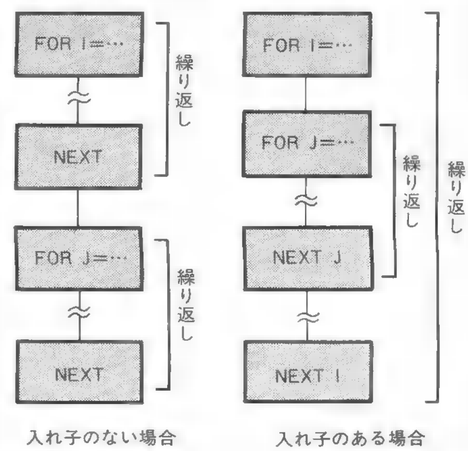
【図123】

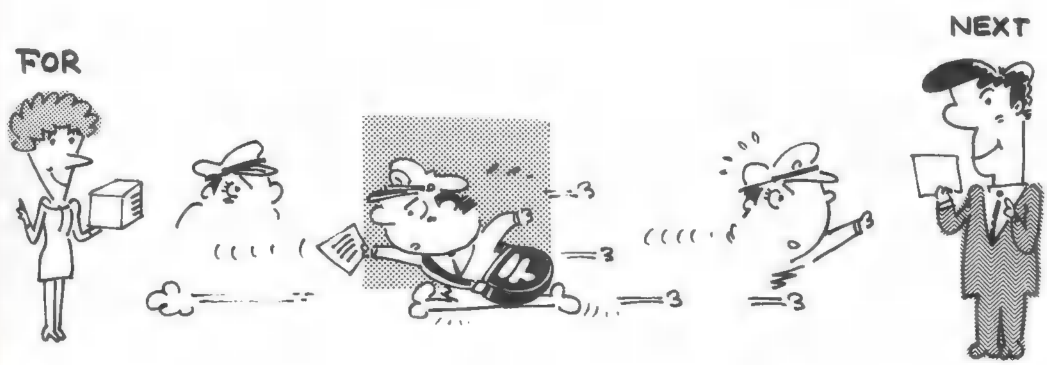
```
 1 ｶｲﾒ ｸﾘｶｴｼ
  2 ｶｲﾒ ｸﾘｶｴｼ
   3 ｶｲﾒ ｸﾘｶｴｼ
    4 ｶｲﾒ ｸﾘｶｴｼ
Ok
■
```

ックされます。超えていなければ，もう1度FOR と NEXT にはさまれた部分を実行します。ここでは，変数Iは1，3，5，7，9と変化し，結局5回ループして11になった時に終了します（図121, 図122, 図123を参照)。

初期値，終り値，増分は負の値や小数点を含んだ値でもよく，増分が＋1の時は省略できます。また，FOR～NEXT の繰り返し部分に別の FOR～NEXT が含まれてもかまいません。これを入れ子ループと呼びます。変数は別なものを使います。入れ子は何重に

【図124】 FOR NEXTの入れ子の有無





なっていてもかまいませんが，入れ子のNEXTが親のループ外にあることは認められません。NEXTの後に，どのFORと対応しているのかを示すため変数を書きます。入れ子がなければNEXTの後の変数は省略できます（図124，図125，図126）。

【図125】

```
10 PRINT "ｶｹｻﾞﾝ 99ﾋｮｳ"
20 FOR I= 1 TO 9
30 FOR J= 1 TO 9
40 PRINT USING"###";I*J;
50 NEXT J:PRINT
60 NEXT I
```

【図126】

```
ｶｹｻﾞﾝ 99ﾋｮｳ
  1  2  3  4  5  6  7  8  9
  2  4  6  8 10 12 14 16 18
  3  6  9 12 15 18 21 24 27
  4  8 12 16 20 24 28 32 36
  5 10 15 20 25 30 35 40 45
  6 12 18 24 30 36 42 48 54
  7 14 21 28 35 42 49 56 63
  8 16 24 32 40 48 56 64 72
  9 18 27 36 45 54 63 72 81
```

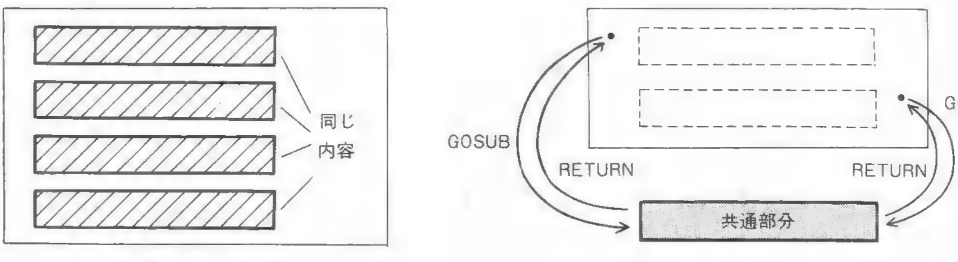
サブルーチン

同じ処理をプログラムのあちこちで必要とする時は，サブルーチンという手法を使います。プログラムのあちこちに同じ命令群を書くのは繁雑ですし，修正が起きた時は，それぞれの修正をしなくてはなりません。そこで，共通に使える命令群を1カ所にまとめ，元のプログラムから必要なつど分岐させます。

この時，元のプログラムをメインプログラム，1カ所にまとめた方をサブプログラムとか，サブルーチンと呼びます。サブルーチンへ分岐して共通処理をした後，元へ戻してやる必要があります。このサブルーチンへ分岐する命令をGOSUB，元へ戻る命令をRETURNといいます（図127）。

GOSUB（ゴーサブ）命令，RETURN（リターン）命令

【図127】 サブルーチンの効果



●重複部分が多く繁雑

●共通部分を別にとり出し，必要な時参照する

GOSUBの後に分岐したい行番号を指定します。RETURNと対になっていることを除くと，GOTOと同じ働きです。

RETURNは本来，GOSUBで分岐した直後に戻りますが，行番号を指定して強制的に戻すこともできます。サブルーチンは必ずRETURN命令で終わらなければならず，GOSUBでサブルーチンへ分岐したのにGOTOで元へ戻ったり，GOTOでサブルーチンへ分岐することは許されません（図128)。

このプログラムでは入力データの範囲チェックを行い，範囲外の時サブルーチンへ分岐して，ブザーを鳴らします。BEEPはブザーを鳴らす命令です。

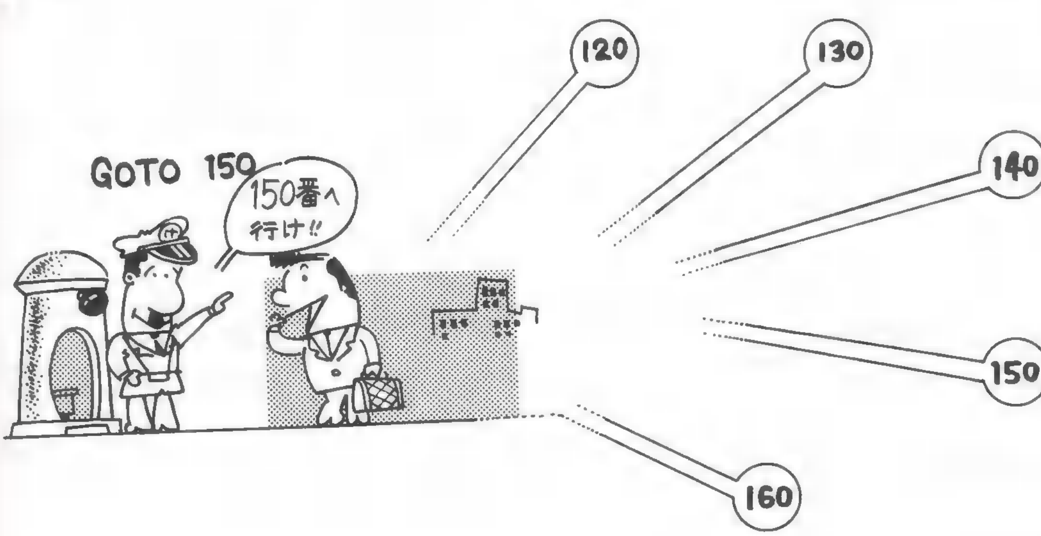
GOTO，GOSUBは，それぞれ無条件に指定の行番号に分岐しますが，条件付きの分岐命令として，ON GOTO，ON GOSUBがあります。

ON GOTO（オンゴーツー）命令

ONの後に書かれた数値の値により，GOTOの後のいくつかの行番号のどれかに分岐します。数値が1の時は，いくつか並んでいる行番号の1番目に，2の時は2番目の行番号にそれぞれ分岐します。数値が0の時および並んでいる行番号の個数よりも大きい

【図128】

```
10 SCREEN 0,0:CLS
20 LOCATE 2,0:PRINT "コード"
30 LOCATE 10,0:PRINT "ショウヒンメイ"
40 LOCATE 30,0:PRINT "スウリョウ"
50 Y=1
60 LOCATE 2,Y:INPUT A
65 IF A=999 THEN END
70 IF A<1 OR A>10 THEN GOSUB 120:GOTO60
80 LOCATE 10,Y:INPUT B$
90 LOCATE 30,Y:INPUT C
100 IF C<0 OR C>100 THEN GOSUB 120:GOTO90
110 Y=Y+1:IF Y<7 THEN 60 ELSE 10
120 FOR I=1 TO 5:BEEP:NEXT
130 RETURN
```



時は，何もせずに次の行へ進みますが，数値の値が負の時はエラーになります。

ON GOSUB（オンゴーサブ）命令

分岐先がRETURNで終わるサブルーチンであること以外はON GOTOと同じ働きをします（図129）。

【図129】

```
10 INPUT "ｸﾌﾞﾝ";K
15 IF K<1 OR K>4 THEN 10
20 ON K GOSUB 100,200,300,30
30 END
100 PRINT "ｸﾌﾞﾝ 1 ﾉ ｼｮﾘ"
110 RETURN
200 PRINT "ｸﾌﾞﾝ 2 ﾉ ｼｮﾘ"
210 RETURN
300 PRINT "ｸﾌﾞﾝ 3 ﾉ ｼｮﾘ"
310 RETURN
```

配　列

プログラムの中で，多くの変数を必要とするとき，それぞれに他のものと区別できるような名前をつけるのは，なかなか大変です。しかし，データの型が同じで使う目的も同じ変数は表形式にまとめ，全体に1つだけ名前をつけて扱うことができます。個々のデータは，その表の中の何番目かという数字で指定されます。この複数の変数を表形式にまとめたものを「配列」といい，配列内の順番を表す数字を「添字」といいます（図130）。

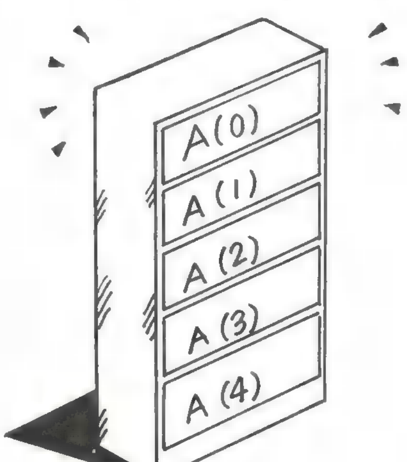
配列名のつけ方は，通常の変数名のつけ方と同じです。添字は，配列名の後に( )で囲んで書きます。添字は変数が使えますから，FOR～NEXTなどを活用すると，すっきりしたプログラムを作ることができます。

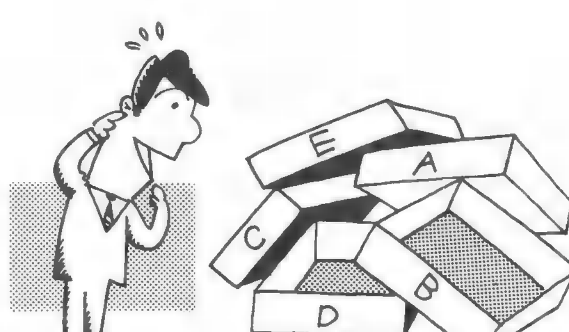
次にバラバラの変数を使った場合（図131）と，配列を使ってまとめた場合（図132）の例をあげます。

2つのプログラムの実行結果はまったく同じですが，配列を使うと同じような命令を何度も書かなくてもすむのがわかります。

添字は必ずゼロから数えます。ただし，例では0番目は使っていません。

表は，必ずしも直線的なものばかりではあ



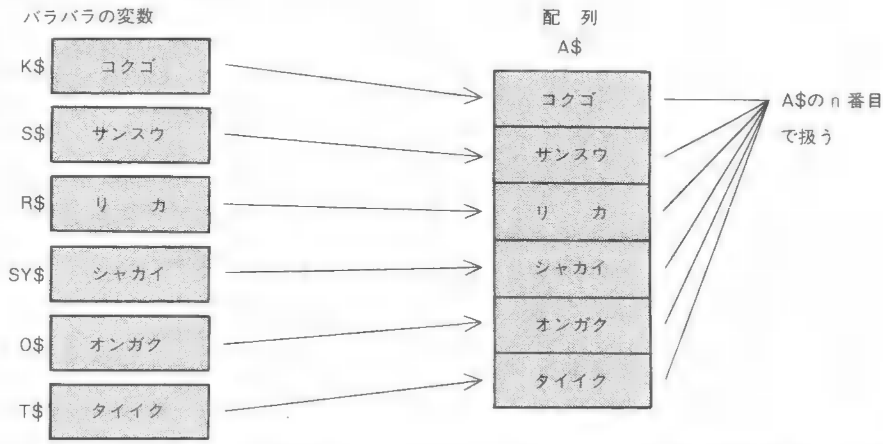


りません。表の5番目の行の左から3番目とか，2枚目の6段目の1番というふうに順番を表す数字，つまり添字が複数個必要となる場合があります。添字の数が1つの時は，一次元配列，2つの時は二次元配列といいます。BASICでは添字は255個まで，つまり255次元まで可能です（図133）。

複数の添字を使う時はレベルの高いものから( )内に書き（,）で区切ります。

DIM（ディメンジョン）命令

【図130】配　列



【図131】

```
10 SCREEN 0,0:CLS
20 INPUT "カモクメイ";A1$
30 INPUT "カモクメイ";A2$
40 INPUT "カモクメイ";A3$
50 INPUT "カモクメイ";A4$
60 INPUT "カモクメイ";A5$
70 INPUT "カモクメイ";A6$
80 CLS
90 LOCATE 10,0:PRINT "1 ";A1$
100 LOCATE 10,1:PRINT "2 ";A2$
110 LOCATE 10,2:PRINT "3 ";A3$
120 LOCATE 10,3:PRINT "4 ";A4$
130 LOCATE 10,4:PRINT "5 ";A5$
140 LOCATE 10,5:PRINT "6 ";A6$
```

【図132】

```
10 SCREEN 0,0:CLS
20 FOR I=1 TO 6
30 INPUT "カモクメイ";A$(I)
40 NEXT
50 CLS
60 FOR I= 1 TO 6
70 LOCATE 10,I-1:PRINT I;A$(I)
80 NEXT I
```

配列をプログラムで使う時は，配列名，配列の次元，データの数を宣言してから使います。配列宣言の命令はDIMといいます。

DIMの後に配列名を書き，その後の( )内に添字の最大の値を書きます。添字が複数の時は，レベルの高い添字から順に，(，) で区切って書いていきます。1つのDIMで複数の配列の宣言をしてもかまいません。

DIM　A$（5），B（10，5，3）

配列宣言をしないで配列を使うと，一次元配列で，添字の最大の大きさは10とみなされます。図132では，添字が6まででしたから配列を宣言しないで使うことができました。

同じ配列を2度宣言するとエラーになります。ループのプログラムを作る場合は，DIMの命令に戻らないように注意します。ただし，CLEAR命令を実行した後は，もう一度配列宣言をしてもかまいません。

DIMが実行されると，すべての変数はゼロまたはヌルストリング（文字変数の場合で空の文字列）になります（図134）。

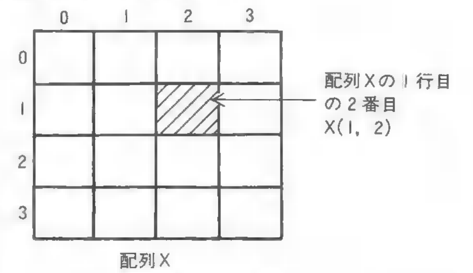
CLEAR（クリア）命令

数値変数をゼロに，文字変数をヌルストリングに初期化し，メモリ領域を設定します。

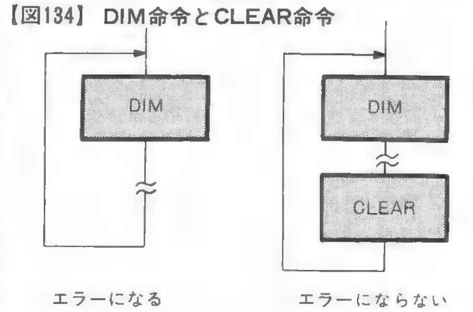
メモリ領域のうち，文字領域として256，BASICが使うメモリの上限として62336が初期設定されていますが，CLEAR命令で変更ができ，一度変更すると次にCLEAR命令で設定をやり直すまで変わりません。

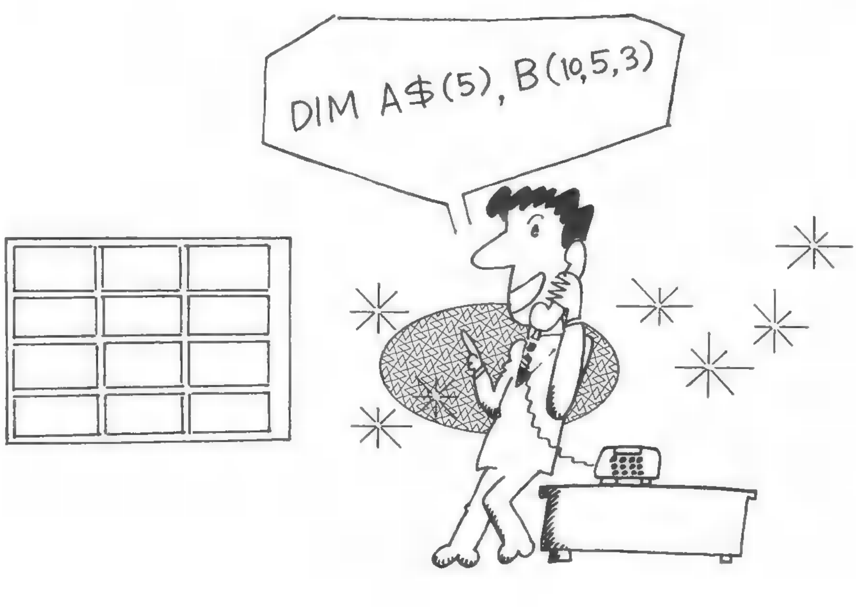
文字領域の変更は，文字型変数の数が多い時，BASICで使うメモリ上限の変更は，機械語プログラムなどを使用の時に行います。

【図133】 複数の添字



【図134】 DIM命令とCLEAR命令





【図135】

```
10 DIM S(100),K#(100)
20 Y=1:GOSUB 210
30 LOCATE 0,Y:INPUT A
40 IF A=999 THEN GOTO 100
50 IF A<1 OR A>100 THEN 30
60 LOCATE 10,Y:INPUT B:S(A)=S(A)+B
70 LOCATE 20,Y:INPUT C#:K#(A)=K#(A)+C#
80 Y=Y+1:IF Y<7 THEN GOTO 30
90 GOTO 20
100 Y=1:GOSUB 210
110 FOR I=1 TO 100
120  LOCATE 0,Y:PRINT I
130  LOCATE 10,Y:PRINT S(I)
140  LOCATE 20,Y:PRINT K#(I)
150 Y=Y+1:IF Y<6 THEN GOTO 190
160 LOCATE 0,7:PRINT "PRESS ANY KEY !";
170 IF INKEY$="" THEN GOTO 170
180 Y=1:GOSUB210
190 NEXT I
200 END
210 SCREEN 0,0:CLS
220 LOCATE 0,0:PRINT "ｺｰﾄﾞ"
230 LOCATE 10,0:PRINT "ｽｳﾘｮｳ"
240 LOCATE 20,0:PRINT "ｷﾝｶﾞｸ"
250 RETURN
```

2つの変更の両方を省略してもかまいませんし，メモリ上限の変更を省略してもかまいませんが，CLEAR で，メモリ上限の変更だけを行うことはできません。

CLEAR 1000，60000

1000：文字領域を256から1000にする

60000：BASICの上限を62336から60000にする

図135に配列を使ったプログラム例をあげます。コードと数量と金額を入力し，コードごとに集計するプログラムです。コードはいま，1～100番とします。

このような場合，もし配列を使わないとすると，数量の合計用の変数が各コード別に100個，金額の合計用の変数が100個必要になります。100個の変数には，それぞれ別の名前をつけなくてはなりません。変数名が違えば，値を代入したり，表示したりする命令も変数名ごとに，それぞれ100回繰り返し書くことになります。

配列を使う場合は，100個の変数を用意してそれぞれに固有の名前をつける代わりに，添字を100にした配列を1個宣言します。

DIM S（100）と書くと，配列Sには0番目から100番目までデータを入れる場所があることになります。また，何番目かを表す添字には変数が使えるので，配列Sの1番目，配列Sの2番目という固定的ないい方だけでなく，配列SのA番目という汎用的な表し方ができます。このため命令は100回書くところを1回で済むわけです。あとは，添字に使われているAという変数に，何番目という適切な数値が代入されればよいことになります。

このプログラムでは，コードを入力してそのコードにより集計するのですから，コード1の時は配列の1番を使って集計，コード2の時は配列の2番目を使って集計ということにします。すなわち，入力されたコードそのものを配列の添字として使うわけです。

ソート

たくさんのデータを扱う時に，それらのデータの順番を処理しやすいように並べかえることがあります。データを並べかえることをコンピュータ用語ではソート（SORT）とか分類と呼びます。ソートでは，通常番号順，日付順など数字でいえば小から大へ並べます。これを昇順ソートと呼びます。アイウエオ順やABC順も昇順ソートの一種として扱います。ソートでは大から小へ並べることがあります。成績の良い順（点数順），金額順などの場合です。これを降順ソートと呼びます。

ソートを行う場合，たくさんのデータをそれぞれ変数に入れていては，プログラムが膨大になってしまいますが，配列を使うと，データ量に関係なく簡単な命令で処理できます。

次にコード順にソートする例をあげます。ソートのスピードを上げるためにプログラム的にはもっといろいろな技法が使われますので，あくまでも一つの例と考えてください（表6，図136）。

ソートすべきデータは，実際はキーボードから入力されたり，別なプログラムで作成さ

【表6】 ソート前とソート後

ソート前

コード	名　前
44	タムラ
30	ヨコヤマ
38	アベ
35	スズキ
48	キムラ

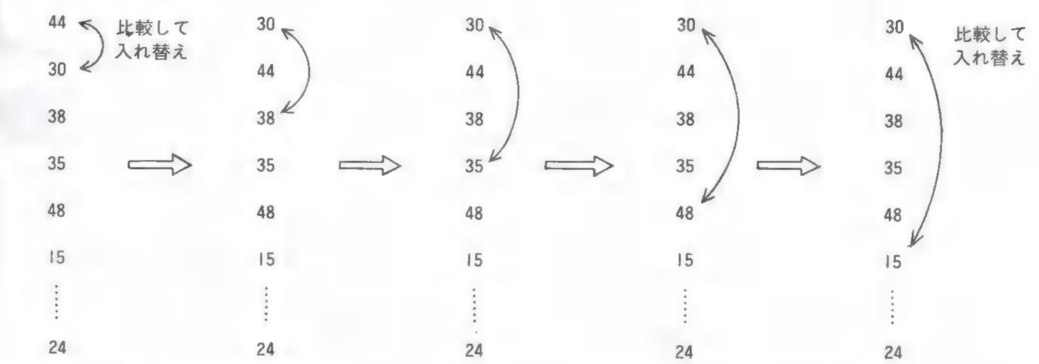
ソート後

コード	名　前
30	ヨコヤマ
35	スズキ
38	アベ
44	タムラ
48	キムラ

【図136】

```
10 DATA 44,ﾀﾑﾗ,30,ﾖｺﾔﾏ,38,ｱﾍﾞ,35,ｽｽﾞｷ,48,ｷﾑﾗ
20 DATA 15,ﾋｸﾞﾁ,34,ｲﾄｳ,50,ｺﾝ,3,ｻﾜｲ,26,ｼﾏﾀﾞ
30 DATA 32,ｱﾗｲ,47,ｴﾄｳ,37,ﾜﾀﾞ,9,ｸﾎﾞ,29,ｾｶﾞﾜ
40 DATA 1,ｵｵﾀ,12,ﾔﾏｸﾞﾁ,20,ﾅｶﾉ,25,ｺﾊﾞﾔｼ,19,ﾆｼ
50 DATA 5,ｲｲﾀﾞ,31,ｻｻｷ,22,ｶﾜｸﾞﾁ,36,ﾀｶｲ,10,ﾌｼﾞﾀ
60 DATA 7,ﾇﾏﾀ,45,ｳｴﾉ,42,ｵｶﾀﾞ,2,ｷﾀ,16,ｼﾐｽﾞ
70 DATA 13,ｲｲｼﾞﾏ,46,ﾀﾅｶ,43,ﾊﾗ,23,ｻｶｲ,11,ｼﾌﾞﾔ
80 DATA 14,ｷｸﾁ,41,ﾄｼﾏ,8,ｿﾉｲ,28,ｺﾝﾉ,17,ﾆｼﾓﾄ
90 DATA 6,ﾏﾂｼﾏ,39,ｵｵｷ,27,ｲｼﾀﾞ,49,ﾀｶﾉ,33,ﾔﾏﾓﾄ
100 DATA 21,ﾉｸﾞﾁ,4,ｻﾜﾓﾄ,18,ｶﾉｳ,40,ﾐﾅﾐ,24,ｾﾉ
110 DIM W(50),W$(50)
120 FOR I=1 TO 50:READ W(I),W$(I):NEXT
130 FOR I=1 TO 49:FOR J=I+1 TO 50
140 IF W(I)<W(J) THEN 170
150 WK=W(I):W(I)=W(J):W(J)=WK
160 WK$=W$(I):W$(I)=W$(J):W$(J)=WK$
170 NEXT J,I
180 FOR I=1 TO 50:PRINT W(I);W$(I);:NEXT
190 END
```

【図137】 ソート



【図138】

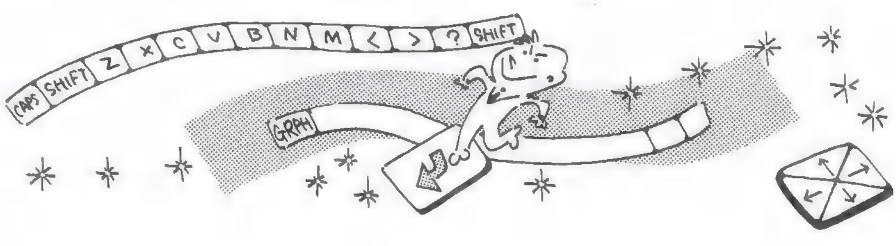
```
1 ｵｵﾀ 2 ｷﾀ 3 ｻﾜｲ 4 ｻﾜﾓﾄ 5 ｲｲﾀﾞ 6 ﾏﾂｼﾏ 7 ﾇ
ﾏﾀ 8 ｿﾉｲ 9 ｸﾎﾞ 10 ﾌｼﾞﾀ 11 ｼﾌﾞﾔ 12 ﾔﾏｸﾞﾁ 1
3 ｲｲｼﾞﾏ 14 ｷｸﾁ 15 ﾋｸﾞﾁ 16 ｼﾐｽﾞ 17 ﾆｼﾓﾄ 18
 ｶﾉｳ 19 ﾆｼ 20 ﾅｶﾉ 21 ﾉｸﾞﾁ 22 ｶﾜｸﾞﾁ 23 ｻｶｲ
 24 ｾﾉ 25 ｺﾊﾞﾔｼ 26 ｼﾏﾀﾞ 27 ｲｼﾀﾞ 28 ｺﾝﾉ 29
 ｾｶﾞﾜ 30 ﾖｺﾔﾏ 31 ｻｻｷ 32 ｱﾗｲ 33 ﾔﾏﾓﾄ 34 ｲﾄ
ｳ 35 ｽｽﾞｷ 36 ﾀｶｲ 37 ﾜﾀﾞ 38 ｱﾍﾞ 39 ｵｵｷ 40
ﾐﾅﾐ 41 ﾄｼﾏ 42 ｵｶﾀﾞ 43 ﾊﾗ 44 ﾀﾑﾗ 45 ｳｴﾉ 46
 ﾀﾅｶ 47 ｴﾄｳ 48 ｷﾑﾗ 49 ﾀｶﾉ 50 ｺﾝ
```

れたファイルであったりします。しかし，ここではソートの機能を知ることが目的なので，データはあらかじめ用意したものを使います。図136では，ソートすべきデータを DATA 命令で定義し，それを READ 命令で，配列に代入します。キーボードや外のファイルから入力されたデータであっても，必ず配列に入れます（図137）。

配列に入れたデータをソートするのに，ここでは1番目のデータS（i）を2〜50番目のデータと比較し，より若い（小さな）コードであれば入れ替えています。1番目のデータが最後まで比較された時，1番目のデータは一番若い（小さな）コードになっています。いわば勝ち抜きトーナメントの勝者です。

次に2番目のデータと，3〜50番目を比較し，同様により若いコードであれば入れ替えます。これを全データ分繰り返すわけです。二重ループ（入れ子）の FOR〜NEXT の部分がこの作業にあたります。

ソートが終わると，実際には，みやすいかたちで印刷したり，ファイルへ出力したりするわけですが，ここでは単純に画面表示しました。実行結果は図138のとおりです。



⑤ 関数の使い方

文字関数の使い方

与えられた文字または文字列の長さを調べたり，分解したりする機能をもつ命令があります。これらは関数と呼ばれ，単独で使うことはできませんが，LET 命令や，IF 命令の中で使用されます。すでに，いくつかの文字関数については説明してきましたが，ここでその文法をまとめてみましょう。

ASC（アスキー）関数

文字または文字列であればその1文字目のキャラクタコードを調べる働きをします。キャラクタコードとは，BASIC で使用しているアスキーコード表（アメリカで制定された標準文字コード）の何番目にあたるかの数字です。パソコン内ではすべての文字は実はこのアスキーのキャラクタコードで扱われています（キャラクタコード表については127ページを参照してください）。

アルファベットの大文字のAは65番目，小文字のzは122番目にあたっています。

次の命令を実行して確かめてください。

PRINT ASC（“A”） ⏎

PRINT ASC（“z”） ⏎

CHR＄（キャラクタダラー）関数

キャラクタコードから，対応する文字を調べる働きをします。ASC 関数の反対の働きです。

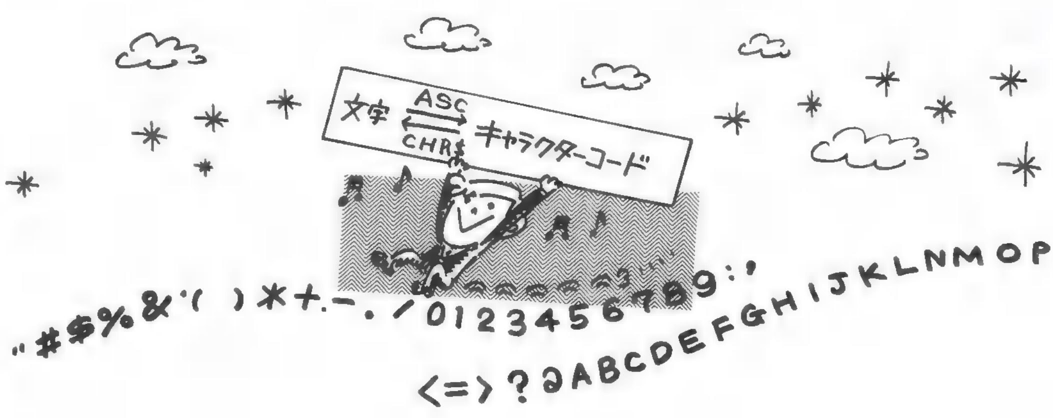
文字 ―ASC→ キャラクタコード
文字 ←CHR＄― キャラクタコード

CHR＄ の後に0～255の数値を指定しますが，その範囲外の数値の場合はエラーとなります。図139のプログラムを実行してみましょう（図140は実行例）。

LEFT＄（レフトダラー）関数，RIGHT＄（ライトダラー）関数，MID＄（ミッドダラー）関数

【図139】

```
10 FOR C=33 TO 122
20 PRINT CHR$(C);
30 NEXT
40 END
```



【図140】

```
Run
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGH
IJKLMNOPQRSTUVWXYZ[¥]^_`abcdefghijklmnop
qrstuvwxyz
Ok
■

 Load"   Save"   Files   List    Run
```

LEFT＄, RIGHT＄, MID＄の関数はそれぞれ，左側から，右側から，まん中から文字列を分解する働きをします。

LEFT＄の後の( )内に，分解する文字列と取り出す長さを指定します。

A＄=LEFT＄（“ABCDE”, 2)

の命令ではA＄には元のデータの左から2文字分ABの2文字が代入されます。

逆に，

B＄=RIGHT＄（“ABCDE”, 2)

では，B＄にDEの文字が代入されます。

MID＄を使って文字列の途中の文字を取り出すときは，( )内に分解する文字列と，文字列の何文字目からと，長さを指定します。

C＄=MID＄（“ABCDE”, 3, 1)

では，C＄には3文字目からの1文字，つまりCが代入されます。

長さが0の時はいずれもヌルストリング(空)が与えられ，長さが文字列（MID＄の場合は残りの文字列）を超えている時は，文字列全部（MID＄の場合は残りの文字列）が与えられます。長さは0～255の範囲内，MID＄の何番目からの数字は1～255の範囲内にないとエラーになります。

文字列の分解を使った図141のプログラムを実行してみましょう（図142)。

VAL（バリュー）関数

VALの後の( )内の文字列の内容が0～9の数字と+－の符号，小数点で構成されている時，これを数値に直す働きをします。文字列の最初の1文字目が文字の時は0の数値となり，途中に文字が入っている時は，それ以降の文字は無視されます。途中の空白は無視されます。

A=VAL（“123.4”)

STR＄（スターダラー）関数

VALの反対の働きをする関数です。数値を文字型に変換します。

数値型 —STR＄→ 文字型
数値型 ←VAL— 文字型

B＄=STR＄（456)

BASICでは，文字型と数値型は厳密に区別され，“580501”のような文字列は，実質

【図141】

```
10 INPUT "ﾋﾂﾞｹ";H$
20 A$=LEFT$(H$,2)
30 B$=MID$(H$,3,2)
40 C$=RIGHT$(H$,2)
50 PRINT "ｼｮｳﾜ"A$"ﾈﾝ"B$"ｶﾞﾂ"C$"ﾆﾁ"
60 END
```

的に数値であっても計算に使うことはできません。しかし，文字列の分解や長さのチェックなど文字列として扱った方が便利な場合もあります。そこで，実質的には数値でありながら文字型として扱ったり，数値に変換したりの機能が必要なわけです。もちろん，アルファベットやカナの文字を数値型に変換しようがありませんから，無理に変換しようとすれば，値はゼロになります。

次の**図143**の例は，**図141**を少し手直ししたものです（**図144**）。

入力された6桁の日付を年，月，日に分解するためには，文字列として扱わねばなりません。分解された，A$，B$，C$ ももちろん文字列です。しかし，05月01日のように，数値の頭にゼロが出力されるのは，あまりみやすいとはいえません。数値の有効数字部分のみを出力して，先頭のゼロを出力しない（ゼロサプレスといいます）ために，PRINT USING を使います。PRINT USING による編集は，数値に対して行うものですから，B$（月が入っている）と C$

【図142】

```
Run
ﾋﾂﾞｹ? 580504
ｼｮｳﾜ58ﾈﾝ05ｶﾞﾂ04ﾆﾁ
Ok
■

Load"  Save"  Files  List    Run
```

【図143】

```
10 INPUT"ﾋﾂﾞｹ";H$
20 A$=LEFT$(H$,2)
30 B$=MID$(H$,3,2)
40 C$=RIGHT$(H$,2)
50 PRINT "ｼｮｳﾜ"A$"ﾈﾝ";
60 PRINT USING"##ｶﾞﾂ";VAL(B$);
70 PRINT USING"##ﾆﾁ";VAL(C$)
80 END
```

【図144】

```
Run
ﾋﾂﾞｹ? 580504
ｼｮｳﾜ58ﾈﾝ 5ｶﾞﾂ 4ﾆﾁ
Ok
■

Load"  Save"  Files  List    Run
```

（日が入っている）を VAL 関数で数値に戻しているわけです。

STRING$（ストリングダラー）関数

指定の1文字または文字列の最初の1文字を任意の数だけ続けた文字列を作ります。STRING$ の後の( )内に，長さと文字を指定しますが，文字の代わりに0～255の範囲の数値が書かれた時は，その数値をキャラクタコードとみなします。

A$=STRING$（8,“*”）

これは，A$=“********”と同じ働きをします。

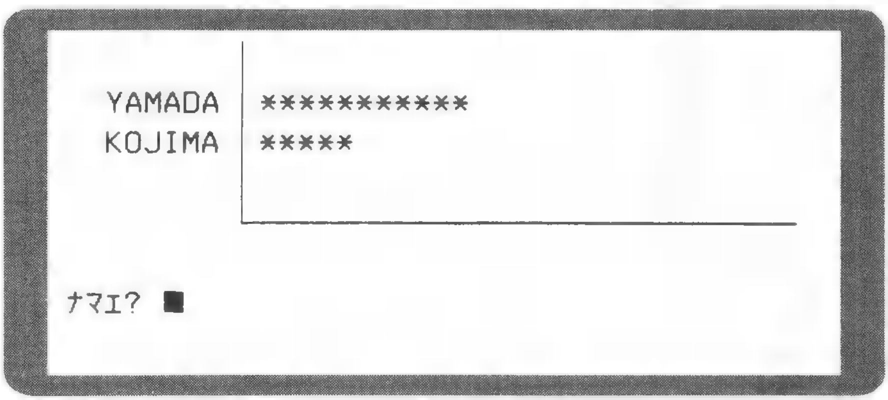
B$=STRING$（3+2, 100）

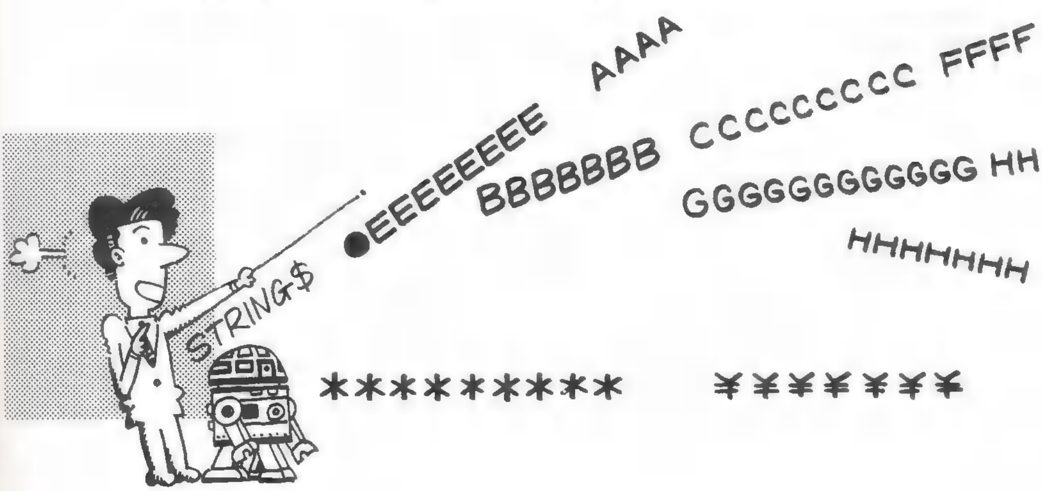
これは，B$=“ddddd”と同じです（図

【図145】

```
10 SCREEN 0,0:CLS
20 FOR I= 0 TO 36:PSET(56,I,1):NEXT
30 FOR I= 56 TO 200:PSET(I,36,1):NEXT
40 FOR Y= 1 TO 3
50 LOCATE 0,6:INPUT "ナマエ";N$
60 LOCATE 2,Y:PRINT N$
70 LOCATE 0,6:PRINT SPACE$(20)
80 LOCATE 0,6:INPUT "セイセキ";A
90 LOCATE 10,Y:PRINT STRING$(A,"¥")
100 LOCATE 0,6:PRINT SPACE$(20)
110 NEXT:LOCATE 0,6
```

【図146】





145, 図146)。

SPACE$ (スペースダラー) 関数

SPACE$ の後の( )内の数値の長さ分の空白を与えます。

A$=SPACE$ (10) は

A$=STRING$ (10, "　")

と同じ働きをします。

TAB (タブ) 関数

TABは, カーソルを水平移動させる働きを持ち, PRINTまたはLPRINTの中でしか使えません。

行の最初の位置を0として指定された位置まで, 空白を出力しながらカーソルを水平移動します。現在のカーソル位置よりも小さい位置を指定した時は, TABの指定は無視されます。

図147のプログラムを実行してTABの働きを確認しましょう (図148)。

【図147】

```
10 PRINT TAB(5) "ABC" TAB(10) "DEF" TAB(5) "XYZ"
20 PRINT SPACE$(5) "ABC" SPACE$(10) "DEF" SPACE$(5) "XYZ"
```

【図148】

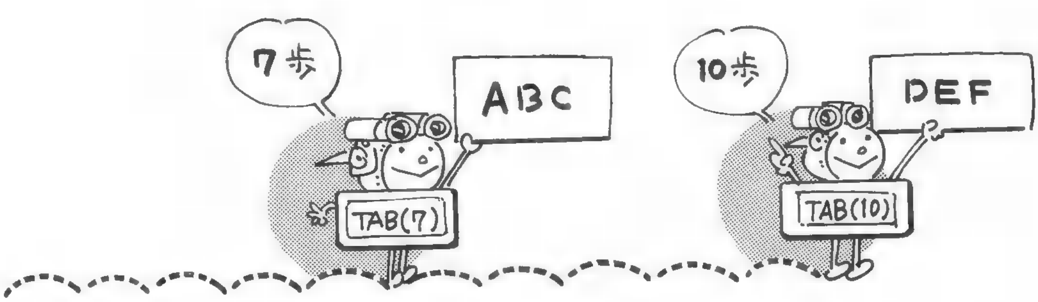
```
Run
     ABC  DEFXYZ
     ABC          DEF     XYZ
Ok
■



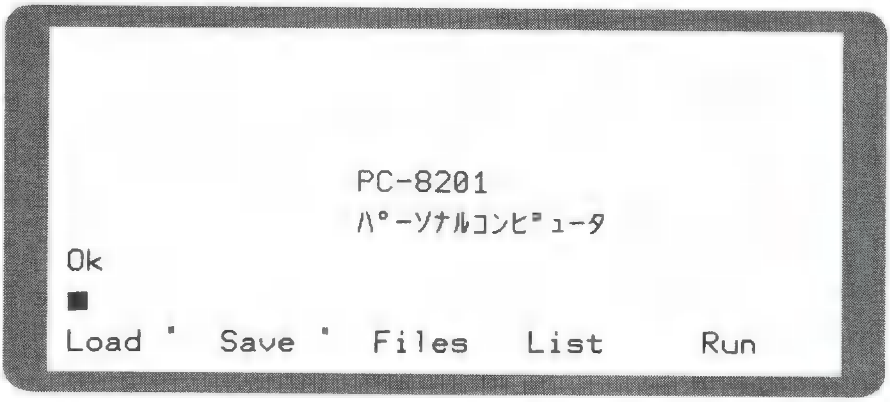
Load "  Save "  Files  List   Run
```

【図149】

```
10 CLS
20 LOCATE 15,4:PRINT "PC-8201";
30 X=POS(1):Y=CSRLIN
40 LOCATE X-7,Y+1
50 PRINT "ﾊﾟｰｿﾅﾙ ｺﾝﾋﾟｭｰﾀ"
```



【図150】



POS（ポジション）関数

現在のカーソルの水平位置を調べます。POS の後の( )内に変数または定数を書きますが，値は何であってもかまいません。

CSRLIN（カーソルライン）関数

現在のカーソルの垂直位置を調べます。

図149のプログラムを実行してみましょう（図150)。Xは22，Yは4になります。

【図151】

```
10 DIM HI(11)
20 K1$="123579"
30 FOR I=0 TO 11:READ HI(I):NEXT
40 INPUT "ｺｰﾄﾞ";C$
50 IF C$="END" THEN END
60 IF LEN(C$)><2 THEN BEEP:GOTO 40
70 C=VAL(C$):IF C=0 THEN BEEP:GOTO 40
80 INPUT "ﾋﾂﾞｹ";H$
90 IF LEN(H$)<>6 THEN BEEP:GOTO 80
100 Y=VAL(LEFT$(H$,2))
110 M=VAL(MID$(H$,3,2))
120 D=VAL(RIGHT$(H$,2))
130 IF Y>58 THEN BEEP:GOTO 80
140 IF M<1 OR M>12 THEN BEEP:GOTO 80
150 IF D<1 OR D>HI(M-1) THEN BEEP:GOTO 80
160 INPUT "ｸﾌﾞﾝ";K2$
170 IF LEN(K2$)><1 THEN BEEP:GOTO 160
180 A=INSTR(K1$,K2$)
190 IF A=0 THEN BEEP:GOTO 160
200 INPUT "ﾅﾏｴ";N$:L=LEN(N$)
210 IF L<1 OR L>14 THEN BEEP:GOTO 200
220 FOR I=1 TO L
230 M$=MID$(N$,I,1)
240 IF M$>=CHR$(166) AND M$<=CHR$(223) THEN GOTO 270
250 IF M$=CHR$(32) THEN GOTO 270
260 SW=1
270 NEXT
280 IF SW=1 THEN BEEP:SW=0:GOTO 200
290 PRINT "ﾆｭｳﾘｮｸ ｻﾚﾀ ﾃﾞｰﾀ ﾉ ﾁｪｯｸ ﾊ ｽﾍﾞﾃ ｶﾝﾘｮｳ ｼﾏｼﾀ"
300 END
310 DATA 31,28,31,30,31,30,31,31,30,31,30,31
```

INSTR（インスター）関数

指定の文字列の中に別の文字列がどの位置にあるかを調べる働きをします。INSTRの後の( )の中に，調べ始めの位置と文字列を2つ書き，前の文字列の中に後に書いた文字列があるかどうかを調べ，あればみつかった位置を，なければ0を値とします。調べはじめの位置は1～255で指定しますが，元の文字列より大きい数字の場合は，関数の値は0となります。調べはじめの位置は省略でき，その時は1とみなされます。

A=INSTR（"ABCXYZ"，"X"）

ABCXYZの中でXを捜し4番目でみつけたので，Aの値は4になります。

いくつかの文字関数の個々の働きはわかっても，実際のプログラムでどのように使うのかは，なかなかマスターできないものです。図151に文字関数の実例として，入力データのチェックを行うプログラムをあげます。入力されるコード，日付，区分，名前が次の条件でなければならないものとします。

- コード　2桁の数字
- 日付　6桁の数字で暦にある日付
- 区分　1桁の数字で，1，2，3，5，7，9のいずれか
- 名前　カナ文字で最大14文字

数値関数の使い方

与えられた数値(引数といいます)に対して，ある一定の演算を行って，その結果を求める機能があります。この機能は数値関数と呼ばれ，N_{82}-BASICには三角関数，平方根などの初等関数があらかじめ組み込まれています。

関数は，単独の命令としては使えませんが，LETやPRINTの中で機能します。また，これらの初等関数は，引数と結果のすべてを単精度として計算しますが，引数に整数しか使えない場合は，小数部分を切り捨ててから計算します。

数値関数の使い方は，関数の後に数値を( )で囲んで書きます。その数値を引数として演算を行った結果が関数の値となります。例えば，数値 A+B の平方根を求める場合は，SQR（A+B）と書きます。SQRは平方根（ルート）計算を行う関数です。

初等関数の数学的な説明，例えばSQRを



使う時の引数は負であってはならないなどについて，およびその活用については，ここでは目的としませんので省略します。N_{82}-BASIC にどのような関数が組み込まれているかのみ説明します。

ABS 関数

絶対値を求めます。

ATN 関数

逆正接（アークタンジェント）を求めます。

COS 関数

余弦（コサイン）を求めます。

EXP 関数

e を底とした指数関数の値を求めます。

FIX 関数

整数部を求めます。

INT 関数

小数点以下を切り捨てて整数化します。FIX と INT の違いは正の数においてはありません。負の数においては，仮に −3.5 を整数化する場合は，FIX（−3.5）では −3 に，INT（−3.5）は −4 になります。

LOG 関数

自然対数を求めます。

SGN 関数

符号を求めます。正の場合は 1，0 の場合は 0，負の場合は −1 になります。

SIN 関数

正弦（サイン）を求めます。

SQR 関数

平方根（ルート）を求めます。

TAN 関数

正接（タンジェント）を求めます。

CDBL 関数

整数または単精度実数を倍精度実数に変換します。

CINT 関数

単精度実数，倍精度実数を整数に変換します。小数点以下は切り捨てます。数値が整数型の範囲（−32768〜＋32767）にない場合はエラーになります。

CSNG 関数

整数，倍精度実数を単精度に変換します。与えられた数値を単精度型に合わせて有効桁 6 桁に四捨五入します。単精度型の範囲（−1.70141E＋38〜＋1.70141E＋38）の範囲内にない場合は，エラーになります。

FRE 関数

現在のバンクのメモリ（RAM）の空きバイト数をチェックします。( )内が数値の場合はテキスト領域（文字領域を差し引いたメモリ領域）の空きバイト数を表します。( )内が文字の場合は，CLEAR 命令で指定した文字領域（初期値は256バイト）のうちの空きバイト数を表します。( )内の数値，文字は何であってもかまいません。

PC-8201は，メニューモードにおいては常に画面右下，BASICモードではBASICモードに入った時の初期画面で，それぞれ，空きバイト数を表示していますが，FRE を使うと，BASIC プログラム内で空き領域を調べることができます。特に，RAM 上にファイルを作成するプログラムにおいて，残りのバイトを調べて警告を出すなどの処理ができます。

```
PRINT  FRE (1); FRE (A$)
```

RND 関数

0 以上 1 未満のランダムな数（乱数）を発生させます。引数の値により乱数の発生は次のように異なります。

- 引数が負の時　新しい乱数系列の設定
- 引数が 0 の時　1 つ前に発生した乱数の値
- 引数が正の時　次の乱数を発生

図152のプログラムを実行すると，次々に乱数が発生されるのを見ることができます（**図153**）。

乱数は0〜0.999999の範囲内にあるため，そのまま使うことはあまりありません。通常は，ある値を掛けて INT で整数化します。いま，乱数に5を掛けると結果は必ず，5> になります。また，100を掛けると必ず，100> になります。どの範囲の数値をランダムに発生させたいかによって，掛ける数を決めればよいわけです。

RND の使い方になれるために，**図154**に数字あてゲームの例をあげておきます。

【図152】
```
10 FOR I=1 TO 4
20 FOR J= 1 TO 4
30 PRINT RND(1);
40 NEXT J:PRINT
50 NEXTI
```

【図153】
```
.505448  .20506   .780736  .571114
.383948  .110289  .962411  .929699
.695605  .545754  .610596  .278547
.209707  .605568  .815212  .639521
```

【図154】
```
10 CLS
20 A=INT(RND(1)*10)
30 PRINT "ｺﾝﾋﾟｭｰﾀ ｶﾞ ｶﾝｶﾞｴﾀ ｽｳｼﾞ(0-9)ｦ ｱﾃﾃ ｸﾀﾞｻｲ":PRINT
35 INPUT "ｽｳｼﾞ";B
40 C=C+1:IF A=B THEN PRINT "ｵｵｱﾀﾘ":GOTO70
50 IF C=5 THEN PRINT "ｻﾞﾝﾈﾝ ﾃﾞｼﾀ" ELSE 35
60 PRINT "ｾｲｶｲﾊ";A;"ﾃﾞｼﾀ"
70 PRINT:C=0:GOTO 20
```

⑥ データファイル

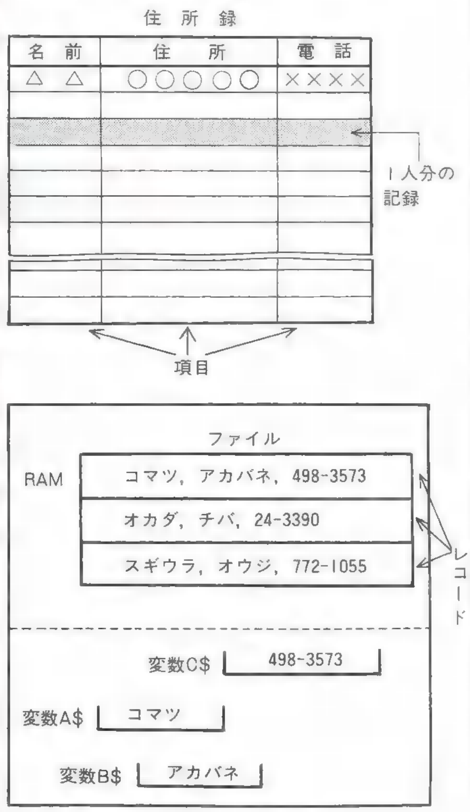
データファイルの概念

データファイルとは，プログラムの中で入力されたデータ，または処理されたデータを，保存のためにまとめたものです。元帳や台帳がコンピュータ処理しやすいように形を変えたものともいうことができます。通常，データファイルを作成するためには，外部記憶装置を必要とするのですが，PC-8201 にはメモリに余裕があればその中にファイルを作成できるという特徴があります。したがって，ここで説明するデータファイルは PC-8201 内つまり RAM 上のファイルです。

ファイルを扱うためにはレコードと項目の意味を理解する必要があります。項目はもちろん1個ずつのデータで，プログラム上は1個の定数または変数がこれにあたります。しかし，実際の仕事上の1件分のデータとは，1個の変数ということはまれで，むしろいくつかの変数で構成されています。

例えば，住所録を考えてみましょう。住所録のデータの1件分とは，名前，住所，電話

【図155】 データファイル



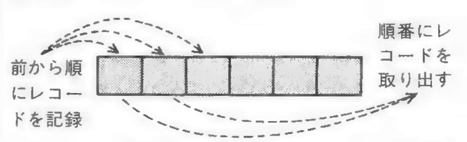
番号などの項目から成り立つはずです。ほかに勤務先とか勤務先電話番号の項目もあるかもしれません。これらのいくつかの項目が集った1件分のデータのことを，コンピュータ用語では1レコード（記録）と呼びます。レコードがたくさん集まったもの，つまり住所録のようなものがデータファイルです（図155）。

ファイル処理には，大きく分けてシーケンシャルな方法とランダムな方法があります。

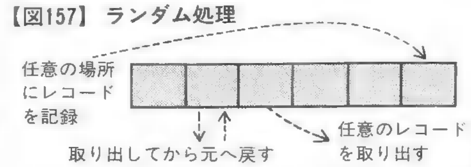
シーケンシャル（SEQUENCIAL）な処理とは，出力された順番にデータファイル上に書き込み，書き込まれた順に入力する処理方法です。ファイル上の一部だけを取り出したり，修正したりすることはできません（図156）。

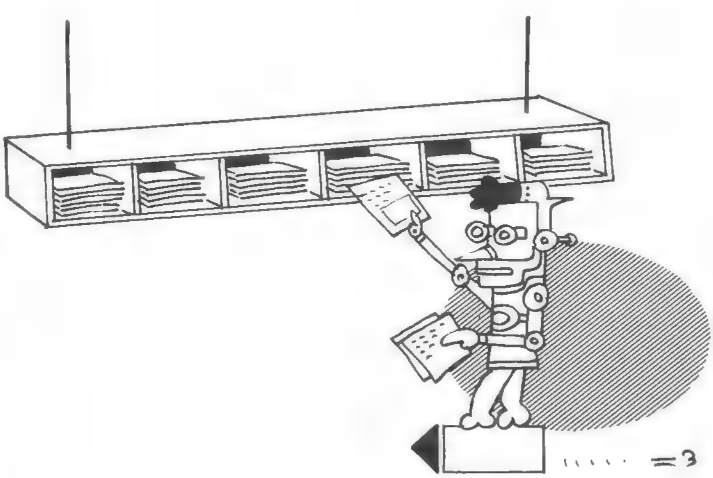
ランダム（RANDOM）な処理とは，出力された順番とデータファイル上の書き込み位置は必ずしも同じではなく，また，入力する時は書き込まれた順番とは無関係に任意の

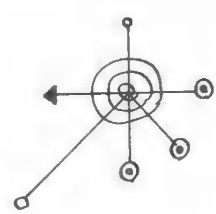
【図156】 シーケンシャル処理



【図157】 ランダム処理







データだけ取り出すことができる処理方法です。1件だけを取り出し修正後元の位置に戻すことも可能です（図157)。

PC-8201のN_{82}-BASICには，シーケンシャルな処理のための命令しか用意されていません。したがって，ランダムな処理は，BASIC上で直接できませんが，PC-8201内蔵のソフトウェアであるTEXTを使うと，簡単に任意のデータを捜したり修正したり，挿入削除をしたりすることができます。

データファイルの作成

すでに説明したようにPC-8201でデータファイルを作成する時は，シーケンシャルな方法で作成されます。実際にデータファイルを作成しながら，必要なBASIC命令を説明しましょう。

データファイルを扱う時は，どんな場合でも最初にOPEN（オープン）命令でファイルを開き，処理の終わりにCLOSE（クローズ）命令で開いているファイルを閉じます。OPENとCLOSEは，ファイル単位に必要です。

OPEN命令では，使用するファイル名とそのファイルの使用目的，および割り当てるファイル番号を宣言します。

使用目的には，入力用（FOR INPUT）と出力用（FOR OUTPUT）と追加用（FOR APPEND）の3種類があり，目的によりOPEN時にBASICが行う作業が違ってきます。入力用の場合は，RAM上のファイルをメモリに読み込んで使うことになるので，指定のファイルがRAM上にあるかどうかを調べます。出力用の場合は，これからRAM上にファイルを書き出すので，新しいファイル名をRAMに登録します。すでに同じ名前のファイルが存在している場合は，新しいファイルのために前のファイルを消します。追加用ファイルは指定のファイルがRAM上にあるかを調べ，みつかったならばその最後のデータの後を出力位置に定めます。それ以外は，出力用と同じように機能します。

ファイル番号とは，使用するファイルごとに1個ずつ割り当てるもので，OPEN以降はファイル名はいちいち指定せず，1番のファイルに書き出す，2番のファイルから読み込むのように，すべてファイル番号で処理します。ファイル番号は，MAXFILES命令で宣言された使用できるファイル数の範囲内で指定します。

OPEN“ファイル名”FOR 目的 AS#ファイル番号で指定します。

OPEN“FILEA” FOR OUTPUT AS #1

ファイル番号について，もう少し詳しく説明しておきます。

プログラムでデータファイルを入出力する時は，必ずバッファ（BUFFER：緩衝）という出入口を通ります。バッファはもちろんメモリ上の領域です。このバッファは全部で15用意されています。ファイル1個につき1つのバッファを割り当てます。したがって，最大15のファイルに同時にバッファを割り当てることができます。このバッファの番号が，実はファイル番号に使われているのです。

しかし，バッファが必要なのは，データの入出力のため，つまりファイルがOPENしている間だけで，CLOSEしてしまえば，他のファイルに割り当ててもよいのです。このため，15のバッファが同時に必要になることはめったにありません。また，バッファ1個につき，267バイトずつRAM領域を使い

ますので，メモリの節約からも不要なバッファは準備しない方がよいということになります。このため準備するバッファの数を宣言するのが MAXFILES の命令です。次の命令では，バッファを3個準備しています。

MAXFILES=3

MAXFILES で指定できるバッファの数は0～15です（図158）。

MAXFILES の命令でバッファの数を指定すると，新たに MAXFILES の命令で指定するまで，バッファの数は変化しません。

いま，バッファの数を3と指定しているとしますと，ファイル番号は必ず1，2，3のいずれかを使わなくてはなりません。また，すでに OPEN 中の他のファイルが使用しているファイル番号を使うこともできません。

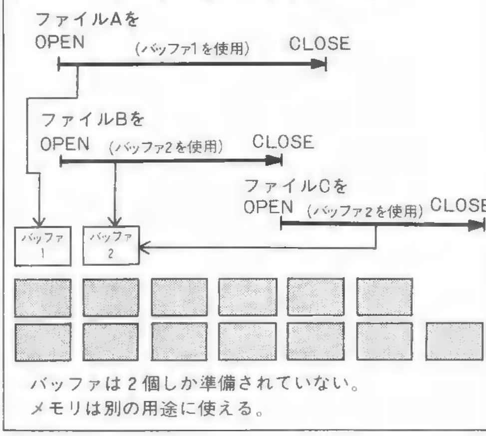
CLOSE 命令は，開いているファイルを閉じ，使用しているバッファとこのファイルとの1対1の関係を解消します。割り当てられているファイル番号はこれ以降，他のファイルに割り当てることができます。

CLOSE の後に閉じたいファイルのファイル番号を書きます。複数のファイル番号を書いて同時に CLOSE することもできます。ファイル番号を省略すると，OPEN 中のすべてのファイルを CLOSE します。

CLOSE #1, #2

CLOSE

【図158】 ファイルとバッファの関連図



#は省略してもかまいません。

実際にファイルへデータを出力する命令は PRINT #です。PRINT #の後にファイル番号を指定し，続いてデータを書きます。データは定数でも変数でもよく，また，1個でも複数でもかまいませんが，できるだけ1件分のレコードは，1つの PRINT #命令の中にまとめた方がプログラム上はすっきりします。

図159はデータファイルへの出力例です。RAM 上には図160のように書き込まれます。

PRINT #の働きは，出力先が RAM 上のファイルであることを除くと，画面表示の PRINT とほとんど同じです。各項目の区切りには，PRINT の場合と同様に(;)と(,)が使えますが，できるだけ(;)を使ってください。(,)を使うと画面上と同じように，14

【図159】

```
10 MAXFILES=1
20 OPEN "keisan" FOR OUTPUT AS #1
30 CLS:PRINT "¥ data output ¥"
40 FOR I= 1 TO 10
50  A=I:B=I^I:C=SQR(I)
60  PRINT A TAB(10) B TAB(25) C
70  PRINT #1,A;B;C
80 NEXT
90 CLOSE:END
```

```
【図160】 1   1   1↵
          2   4   1.41421↵
          3   27   1.73205↵
          4   256      2↵
          5   3125      2.23607↵
          6   46656      2.44949↵
          7   823544      2.64575↵
          8   1.67772E+07      2.82843↵
          9   3.87421E+08      3↵
          10   1E+10   3.16228↵
```

桁ずつ区切って出力されるため余分なスペース分だけファイルが大きくなります。

文字型データを扱う時は，文法上の各項目の区切りとは別に，データ上の区切りが必要となります。次の2行の命令を実行してみてください。

PRINT　123；456 [↵]

PRINT　“A”；“B” [↵]

同じ(；)を区切りに使っても数字データと違い，文字データでは完全に2つのデータは連続してしまいます。仮に空白を入れても，空白も文字列の一部とみなされるため，複数の文字データも一つとして扱われてしまいます。

PRINT　“A”，“B” [↵]

PRINT＃は，画面上にみえないだけでPRINTと同じように働きますから，文字データの区切りに(；)を使った場合，項目やレコードの区切りがなくなって全部のデータが1つに連続してしまいます。その結果，このファイルからデータを読み込む時，本来別々の文字データが1つにまとまって，入力され，はなはだ具合の悪いことになります。データの件数や型がファイル作成時とは変わってしまい，入力の途中でエラーが発生することはまちがいありません。

そのため，文字データを出力する場合は，区切り自体をファイルに書き込むようにします。

PRINT＃1，“AB，”；“CD”

この例の場合，ファイル上は，AB，CDと書き込まれるため，入力する時も(，)を区切りとしてABとCDを別々の項目として扱うことができます。変数の場合は次のように書きます。

【図161】

```
10 OPEN "telF" FOR OUTPUT AS #1
20 SCREEN 0,0:CLS
30 PRINT TAB(10) "※ ｼﾞｭｳｼｮﾛｸ ※"
40 PRINT "ﾅﾏｴ" TAB(10) "ｼﾞｭｳｼｮ" TAB(25) "ﾃﾞﾝﾜ"
50 Y=2
60 LOCATE 0,Y:INPUT A$
70 IF A$="end" THEN CLOSE:END
80 LOCATE 10,Y:INPUT B$
90 LOCATE 25,Y:INPUT C$
100 PRINT #1,A$;",";B$;",";C$
110 Y=Y+1:IF Y<7 THEN 60 ELSE 20
```

PRINT　#1, A$; ","; B$

または

PRINT　#1, A$ "," B$

文字データをファイルに出力するプログラム例を図161にあげます。名前，住所，電話番号をキーボードから入力し，telF というRAM上のファイルへ出力するものです。

このプログラムを実行し，データを数件入力して，RAM上にファイルを作成してください。

データファイルの読み込み

既存のデータファイルを扱う場合も，必ず最初と最後にOPENとCLOSEの命令が必要です。OPENの使用目的は入力用（FOR INPUT）でなければならず，RAM上に指定のファイルがなければ，?FF ERROR IN 行番号（行番号で指定のファイルがみつからないの意味）のエラーが表示され，プログラムは中断されます。

データファイルからデータを読み込む命令はINPUT#です。キーボードから変数に値を入れるINPUT命令とほぼ同じ働きをします。

INPUT#1, A, B, C

INPUT#の後に続く数字はファイル番号で，その後の変数名はいくつ書いてもかまいません。しかし，1レコード分ずつ入力した方が，プログラム上はわかりやすくなります。

図162に入力プログラムの例を示します。ファイルから10レコード読んで表示します。

データ件数があらかじめわかっていないとこのようなプログラムは作れません。書き込まれているデータ件数を超えて入力しようとすると，ファイルが終わった（END OF FILE）を意味するエラー表示が出て，処理は中断されます。

?EF　ERROR　IN　行番号

入力プログラムでは，読みすぎを起こさないように，ファイルの終わりをチェックする必要があります。ファイルの終わりをチェックする関数にEOFがあります。EOFの後の( )内にファイル番号を書いて，どのファイルのチェックかを指定します。EOF関数はファイルの最後のデータを読むと論理式上の真になります。具体的には，−1の値をとります。最後のデータの1つ前までは偽，具体的には0の値です。したがって，ファイルの終わりかどうかを次のようにチェックします。

IF　EOF（ファイル番号）THEN～

または

IF　EOF（ファイル番号）＝−1 THEN～

図163に，RAM上のtelFを読み込み画面表示するプログラム例を示します。

ファイル処理においても，キーボード入力のLINE INPUT命令に似たLINE INPUT#命令が使えます。この命令はファイルから読み込む時，[⏎]キー（キャラクタ

【図162】

```
10 OPEN "Keisan" FOR INPUT AS #1
20 FOR I= 1 TO 10
30 INPUT #1,A,B,C
40 PRINT A;B;C
50 NEXT
```

【図163】
```
10 OPEN "telF" FOR INPUT AS#1
20 PRINT TAB(5)"XX  TELEPHONE  XX"
25 IF EOF(1) THEN CLOSE:END
30 INPUT #1,X$,Y$,Z$
40 PRINT X$ TAB(10) Z$ TAB(25) Y$
50 GOTO 25
```

コードの13）までを1行として，区切らずに変数へ代入します。RAM上のプログラムファイルをデータとして読み込む時などに活用します。第4章のTELCOMに具体例がありますので参照してください。

キーボードから，指定された長さ分の文字の入力を行うINPUT関数は，ファイル番号を指定すると，ファイルからの入力として使うことができます。

INPUT＄（5）　←キー入力

INPUT＄（5, #1）　←ファイル入力

ファイル番号

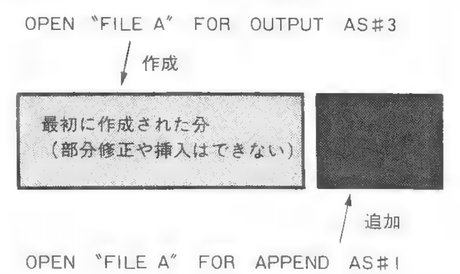
この命令は，ファイル内から1文字ずつデータを取り出してチェックする時に便利です。ファイルの読み込みでは，入力した文字を画面に表示しません。

データファイルへの追加

作成済みのファイルへデータを追加する場合，シーケンシャルな処理では，その一番最後に追加していくことができます（図164）。

追加を行う場合は，OPENにおける使用目的を追加用（FOR APPEND）にします。追加を行う時に，指定したファイルがRAM上に存在していないと，もちろんエラーになります。最初に既存のファイルを捜し，その一番最後からデータの出力を開始することを除くと，出力用とまったく同じ働きをします。プログラム的には，OPEN命令の使用目的を，APPENDにする以外は，出力用のプ

【図164】 データファイルの追加





ログラムと同じです（図165）。

実際には，最後に追加する以外の追加を行いたい場合があります。2番目と3番目のデータの間に1件新しいデータを挿入したい場合などです。しかし，シーケンシャルな処理方法では，挿入，削除，修正のいずれもできません。しかし，元のファイルから，別の新しいファイルを作り直しするならば，ファイル内のデータに対し，この挿入，削除，修正が可能になります。

プログラム例を図167に示します。この方法の場合，RAM上に新旧2つのファイルが存在してしまいます。すぐ古い方は消すとしても，作成中にRAMが一杯になってしまう危険性があります。

また，新旧2つのファイルは同じファイルの手直ししたものなのですが，RAM上に一緒に存在する場合同一の名前は使えませんから，別々の名前をつけます。そのため，ファイル名がどんどん変化してしまうという欠点があります。新しいファイルにはとりあえず仮の名前をつけておき，古いファイルを消した後，新しいファイルの名前を元の名前に変更するという方法もありますが，煩わしいことに変わりはありません。

このプログラムでは，telFという名のファイルからデータを1件ずつ入力し，画面に表示して処理方法をたずねます。挿入と答えると，新しいデータの入力を求め，入力されたデータは画面に表示されているデータより先に新しく作成するnewFへ出力します。その後，もう一度，画面上に出ているデータの処理方法をたずねます。

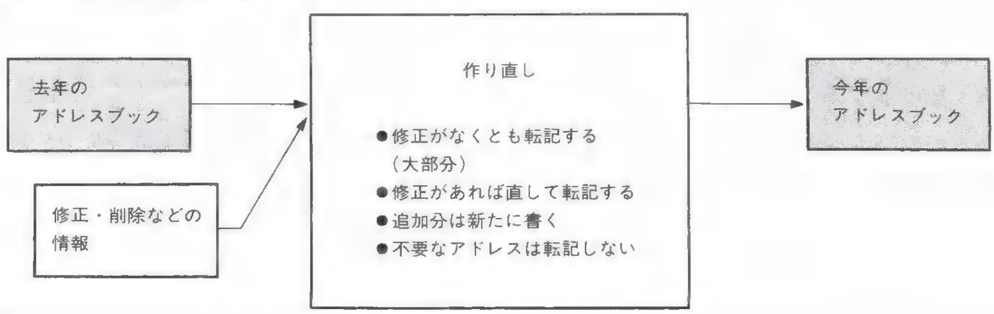
削除と答えると，何もせず次のデータの入力へ進みます。newFへ転記しないのです。

修正と答えると，新しいデータの入力を求

【図165】

```
10 OPEN "telF" FOR APPEND AS #1
20 SCREEN 0,0:CLS
30 PRINT TAB(10) "* ｼﾞｭｳｼｮﾛｸ *"
40 PRINT "ﾅﾏｴ" TAB(10) "ｼﾞｭｳｼｮ" TAB(25) "ﾃﾞﾝﾜ"
50 Y=2
60 LOCATE 0,Y:INPUT A$
70 IF A$="end" THEN CLOSE:END
80 LOCATE 10,Y:INPUT B$
90 LOCATE 25,Y:INPUT C$
100 PRINT #1,A$;",";B$;",";C$
110 Y=Y+1:IF Y<7 THEN 60 ELSE 20
```

【図166】 ファイルの作り直し



【図167】

```
10 MAXFILES=2
20 OPEN "telF" FOR INPUT AS #1
30 OPEN "newF" FOR OUTPUT AS #2
40 SCREEN 0,0:CLS
50 PRINT TAB(10) "※※  TELEPHONE  ※※"
60 PRINT "ﾅﾏｴ" TAB(10) "ｼﾞｭｳｼｮ" TAB(25) "ﾃﾞﾝﾜ"
70 IF I=1 THEN 100
80 IF EOF(1) THEN GOTO 260
90 INPUT #1,A$,B$,C$
100 LOCATE 0,3:PRINT A$
110 LOCATE 10,3:PRINT B$
120 LOCATE 25,3:PRINT C$
130 LOCATE 0,5:PRINT "ｿｳﾆｭｳ=1 ｻｸｼﾞｮ=2 ｼｭｳｾｲ=3 ｿﾉﾏﾏ=4"
140 LOCATE 0,6:INPUT "ﾊﾞﾝｺﾞｳ ﾃﾞ ｾﾝﾀｸ";I
150 ON I GOTO 170,40,170,240
160 GOTO 130
170 LOCATE 0,5:PRINT "ｱﾀﾗｼｲ ﾃﾞｰﾀ ﾉ ﾆｭｳﾘｮｸ" TAB(15)
180 LOCATE 0,6:PRINT SPACE$(40);
190 LOCATE 0,2:INPUT X$
200 LOCATE 10,2:INPUT Y$
210 LOCATE 25,2:INPUT Z$
220 PRINT #2,X$ "," Y$ "," Z$
230 GOTO 40
240 PRINT #2,A$;",";B$;",";C$
250 GOTO 40
260 CLOSE
270 KILL"telF.DO":NAME"newF.DO" AS "telF.DO"
280 END
```

め，これを出力した後，次のデータの入力へ進みます。画面表示されているデータについては修正済みのものが出力されたので不要になったわけです。

そのままと答えると tel F のデータをそのまま new F へ出力します。

このように，まったく新しいファイルを別に作り直す時に，不要なものを転記しなかったり，新しいものを追加したりすることで，削除・挿入を行うわけです。ただし，データの件数が多く，しかも，修正がたった1件というような時は，非常に無駄の多い方法ということになります。

PC-8201 のシステムでは，RAM 上に同一ファイル名が存在することができません。tel F と new F は同じアドレスブックの新旧ですが，プログラム上はこのように別ファイル名にしておきます。完全に新しいファイルが作成された後は，旧ファイルは不要になりますから，消してしまいます。その後，新しく作ったファイルを仮の名 new F から正規の名前 tel F に変更しておきます。

図167の最後に使用されている KILL はファイルを消す命令，NAME はファイル名を変更する命令です。メニューモードにおける KILL と NAME も参照してください。

KILL（キル）命令

指定したファイルを RAM 上から削除する働きをします。拡張子もつけて指定します。

KILL "new F. DO"

現在処理途中にあるファイル，すなわちOPEN されたままになっているデータファイル(.DO) と，アクセス中のファイル，すなわち RAM 上にセーブののち，BASIC 領域に呼び出されている .BA ファイルについては，削除できません。

NAME（ネーム）命令

RAM 上にあるファイルのファイル名を変更します。旧ファイル名と新ファイル名共に拡張子もつけて指定します。

NAME "OLD. BA" AS "NEW. BA"

NAME に続けて現在あるファイル名を書きます。RAM 上に存在しないファイル名を指定するとエラーになります。旧ファイル名の後に AS と新ファイル名を書きます。新ファイル名は，RAM 上にすでに存在しているファイルと重複してはなりません。また，新旧の２つのファイルの拡張子は一致していなければなりません。いいかえると，拡張子の変更はできません。

N_{82}-BASIC ではシーケンシャルファイルしか扱うことができず，シーケンシャルファイルでは直接ファイルの挿入，削除，修正の処理はできないことを述べてきました。新しいファイルへ転記する方法をとれば，挿入，削除はできますが，効率的には問題があります。

PC-8201 の TEXT ソフトウェアでは，RAM 上のファイルの編集が簡単にできますから，BASIC で使うファイルも，TEXT でメンテナンス処理するのが一番簡単でしょう。

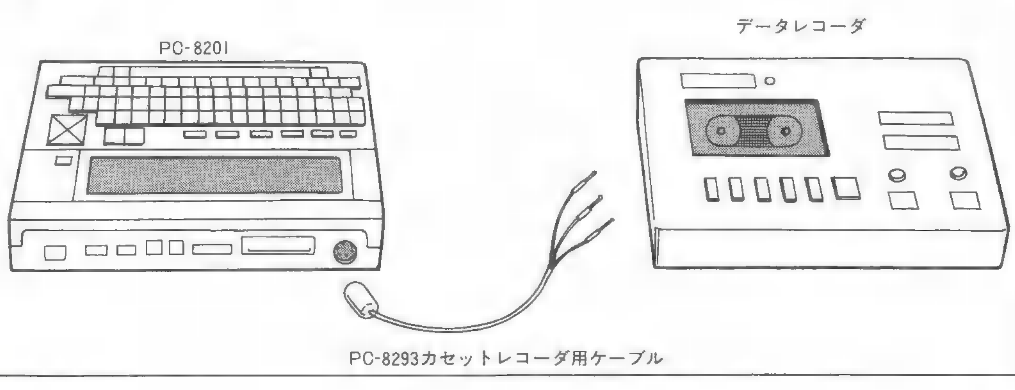
7 カセットテープの入出力

カセットレコーダの接続

PC-8201 には，PC シリーズのカセットレコーダはもちろん，音楽用のカセットレコーダのほとんどのものが接続できます。

PC-8201 は，実行中のプログラムとは別に，プログラムやデータファイルを自分自身の空き領域に保存しておけます。その意味では，外部記憶装置は不要ですが，メモリ容量は無限ではありませんし，大事なプログラムやデータファイルをあやまって消してしまうといった事故対策のうえでも，メモリ以外の装置にプログラムやデータファイルを保存しておく必要がでてきます。

【図168】 データレコーダとの接続



ここでは，カセットレコーダを外部記憶装置として使い，プログラムやデータファイルを記録する方法を説明します。

まず，PC-8201の付属品のPC-8293カセットレコーダ用ケーブルを使ってカセットレコーダとPC-8201を接続します（図168)。

カセットレコーダを接続する時は，ケーブルの赤いプラグをCMT. INまたはマイク端子（入力用）へ，白いプラグをCMT. OUTまたはイヤホーン端子（出力用）へ，黒いプラグをリモート端子へ接続します。カセットレコーダの種類によっては，リモート端子を接続できないものがあります。リモート機能がなくともプログラムのロードやセーブは可能ですが，操作が少しめんどうになります。

接続の後，ON/OFFスイッチのあるものはスイッチをONにして，カセットレコーダが使える状態にします。カセットテープをセットして，セーブは録音と同じようにRECとPLAYのボタンを，ロードは再生と同じようにPLAYのボタンを押します。カセットレコーダはPC-8201の制御を受けてますので，ボタンを押してもテープは止まっています。このようにあらかじめ作動状態にしておいてBASICからテープに対する読み込み，書き込みの命令を実行すると，自動的にテープが動き，命令が終わると自動的に止まります。

早送りや巻き戻しなど入出力とは別にテープだけ動かしたい場合は，カセットレコーダのモータをONまたはOFFにします。モータをONにしてテープを動かしたい時は，[M] [O] [T] [O] [R] [1] [⏎] です。MOTORの後の数字は1以外でも1～256の値であれば何でもかまいません。モータをOFFにしてテープを止めたい時は，[M] [O] [T] [O] [R] [0] [⏎] です。

カセットレコーダにリモート端子がない場合は，テープを動かしたり止めたりする機能が働きませんから，すべて手動で行います。

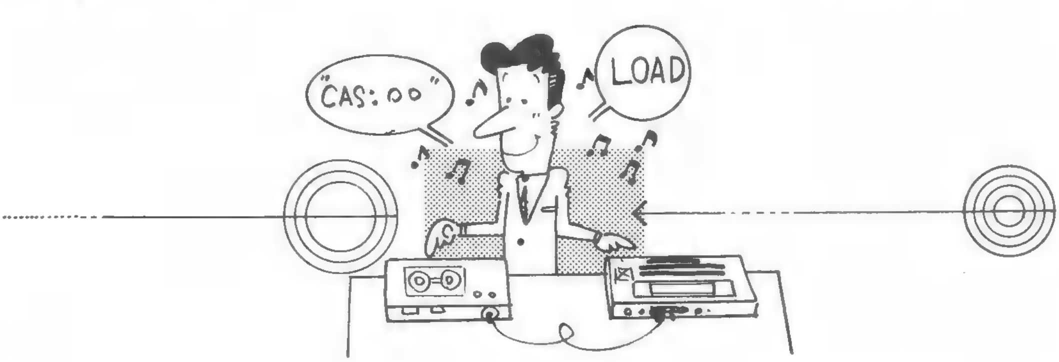
プログラムのロードとセーブ

プログラムをカセットからロードしたり，カセットへセーブしたりする時は，PC-8201では，BASIC上で行う場合と，メニュー上で行う場合とがあります。BASIC上で行えば，BASIC領域にあるプログラムのロードおよびセーブとなり，メニュー上であれば，RAM上のファイルのロードおよびセーブになります。

まず，BASIC上でのロードから説明します。

プログラムの入ったカセットテープをカセットレコーダにセットし，再生状態にします。ロードの命令は2種類あります。

LOAD “cas：プログラム名”



CLOAD “プログラム名”

どちらを使ってもかまいません。プログラム名は，6文字以内で指定します。プログラム名を省略すると，テープ上の最初のプログラムがロードされます。

リモート端子が接続されていると，カセットレコーダを再生状態にしただけでは動きませんが，命令を入力し，[⏎] キーを押すと動きはじめます。

該当のプログラムがみつかると，画面にみつかった（FOUND）と表示し，ロードが完了すると OK がでます。テープ上に複数のプログラムが記録されていて，求めるプログラムが後の方にある時は，途中のプログラムは SKIP の表示だけで読み飛ばされます（図169）。この場合，適当なカウンタまでテープを早送りしてからロードさせてもかまいません。

ロードが完了するとカセットレコーダは自動的に止まります。ロードの途中で止めたい時は [SHIFT] キーと [STOP] キーを同時に押します。カセットのような周辺装置を止めるには [STOP] キーだけでなく，必ず [SHIFT] キーと併用します。

カセットからのロードも，RAM からのロードと同じようにメモリ内のプログラムを消してから行われるので，NEW 命令で消してからロードする必要はありません。

次に，BASIC 上でのセーブを説明します。

まず LIST 命令によりセーブすべきプログラムが BASIC 領域にあるのを確認します。次にカセットレコーダにテープをセットします。テープの最初からセーブするときは，再生状態にして，[M] [O] [T] [O] [R] [I] [⏎] と入力します。再生状態にしただけではテープは止まっていますが，MOTOR 1 の命令で動き出します。続いて [M] [O] [T] [O] [R] [0] [⏎] と入力します。これでテープは止まります。

タイプしている間にカセットレコーダのカウンタは3位進んでいるはずです。テープの最初の磁化されていない部分を巻くための操作で，3～5位を進めます。手でテープを巻いてもよいのですが，以前の記録内容が残っていることもありますから，録音状態でテープを進めた方がよいでしょう。

テープにすでに何本かのプログラムがセーブされていて，そのプログラムを消したくない場合は，カウンタで確認しながら，テープをその位置まで早送りします。その後，録音状態にします。セーブもロードと同じように2種類の命令があります。

SAVE “cas：プログラム名”

CSAVE “プログラム名”

いずれも BASIC 領域にあるプログラムをバイナリー形式（圧縮した形）でテープへ記

【図169】

```
Load "cas:EX-5"
Skip:EX-1
Skip:EX-2
Found:EX-5
Ok
■

Load "  Save "  Files  List  Run
```

録します。圧縮しない形，すなわち，LIST命令でみると同じ形のプログラムをアスキー形式といいます。アスキー形式でセーブしたい場合は次のように書きます。

SAVE “cas：プログラム名”, A

CSAVE にはアスキー形式でセーブする機能はありません。

アスキー形式でセーブすると時間が余分にかかり，テープも長く必要とするので，通常はバイナリー形式（何も指定しない）でセーブしますが，他のPCシリーズのパソコンでロードさせるプログラムや，MERGE命令で使うプログラムはアスキー形式でセーブしておく必要があります。

カセットへのセーブを行った場合は，RAMへのセーブと違い，まったく同じコピーが作られ，メモリに残っているプログラムは何ら変化しません。もちろん，メニューに戻してもBASIC領域はそのままです。

SAVEまたはCSAVEの命令を正しく入力し，［⏎］キーを押すと，書き込みがはじまり，終わるとOKが画面表示されます。テープは自動的に止まります。

カセットからのロードでは，プログラムが正しくロードできるまで，OKの状態にはなりませんが，カセットへのセーブの場合は，カセットレコーダがセーブ状態でなくても一方的に出力をはじめ出力が完了すると，OKが表示されます。セーブ状態にないというのは，①PC-8201にカセットレコーダが接続されていない，②カセットレコーダの電源がOFFになっている，③録音ボタンが押されていない，④テープが先頭のセロファン部分などの場合です。そのほかの操作ミスやカセットレコーダの故障の場合でも，セーブ上はエラーになりませんので注意が必要です。

セーブが正しく行われたか，つまりテープ上のプログラムは，BASIC領域にあるプログラムとまったく同じかをチェックする命令にCLOAD?があります。

CLOAD?“プログラム名”

カセットレコーダを再生状態にし，この命令を入力すると，テープ上に指定のプログラムを捜し，みつかった（FOUND）ならば読み込みながら，メモリ内のプログラムをチェックします。プログラムの最後まで，完全に同じものであれば「Ok」，違っていれば「Bad」と表示します。CLOAD?を実行しても，メモリ内のプログラムはまったく変化しません(図170)。

メニュー上で行うロードとセーブは，RAM上のファイルにつまりプログラム，データファイルの両方に対して行います。まず画面をメニューモードにします。

ファンクションキー1番［f·1］を押すと，「Load from」が表示されます（図171）。

【図170】

```
Save "cas:EX-6"
Ok
cload?"EX-6"
Ok
■



Load "  Save "  Files  List    Run
```

ロードすべきファイルの名前を入力します。CAS：は省略できます。ファイル名も省略すると，テープ上の最初のファイルが対象になります（図172)。

次に「Save as」が表示されます（図173)。

ロードされたファイルは RAM 上に記憶されるので，その時のファイル名を入力します。このファイル名には，必ず拡張子をつけます。バイナリー形式のプログラムには，.BA を，アスキー形式のプログラムおよびデータファイルには .DO を，機械語プログラムには .CO をつけます。

このファイル名が RAM 上で新しいものならば，「Ready?」が，現存のものと同じならば「Sure?」が，ファイル名の直後に表示されます（図174)。

いずれも Y と入力すると，ロードを開始します。READY に対しては ⏎ のみでも働きます。Y 以外を入力すると，LOAD 命令自体がキャンセルされます。

【図171】

```
1983/05/04 15:20:18     (C) Microsoft #1
 BASIC     TEXT      TELCOM     EX-1.BA
 EX-2.BA   -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
Load from _
```

【図172】

```
1983/05/04 15:20:32     (C) Microsoft #1
 BASIC     TEXT      TELCOM     EX-1.BA
 EX-2.BA   -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
Load from EX-3
```

【図173】

```
1983/05/04 15:21:03     (C) Microsoft #1
 BASIC     TEXT      TELCOM     EX-1.BA
 EX-2.BA   -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
 -.-       -.-       -.-        -.-
Save as _
```

SURE? の場合は新しいファイルのために元のファイルを消してからロードを開始しますので注意してください。

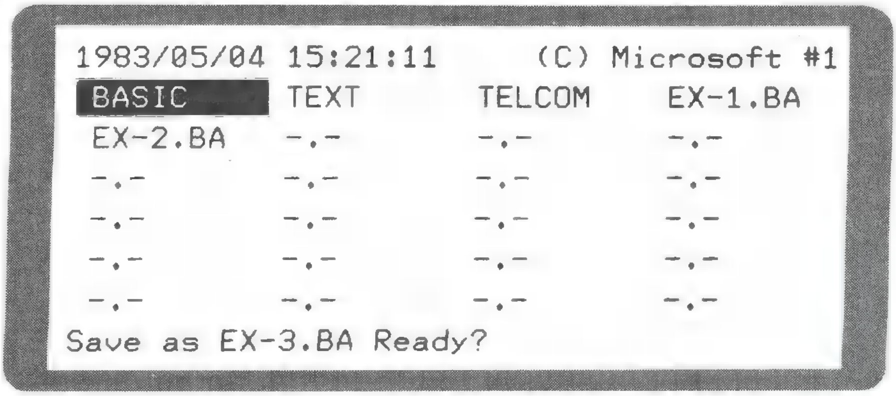
ファイルがみつかると「Found」を，みつかるまえに別のファイルがある場合には「Skip」を，画面の最下段に表示します。ロード完了すると，メニュー画面に新しいファイル名を表示して終わります（図175）。

メニュー上でセーブを行う時は，カーソルをセーブしたいファイル名に移してから，ファンクションキー2番 f·2 を押します。「Save ファイル名 as」が表示されます（図176）。

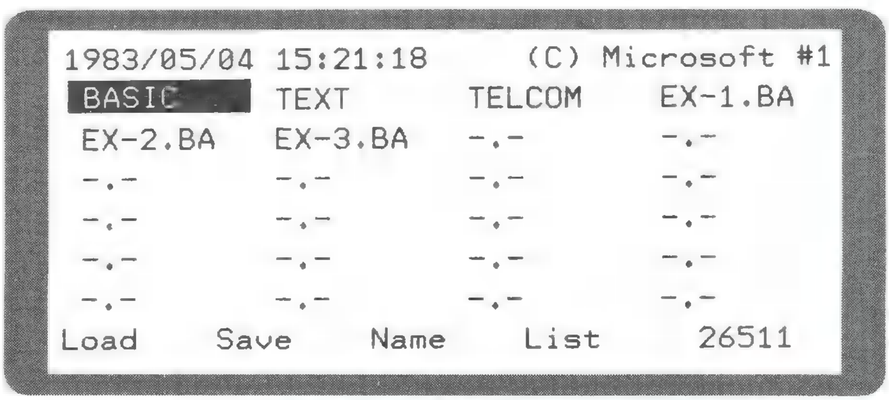
CAS：ファイル名を入力します。CAS：は省略できます。ファイル名を省略するとRAM上のファイルと同じ名前がつきます（図177）。

続いてテープに書き込む方法は，バイナリー形式かアスキー形式かを聞いてくるのでBまたはAで入力します（図178）。

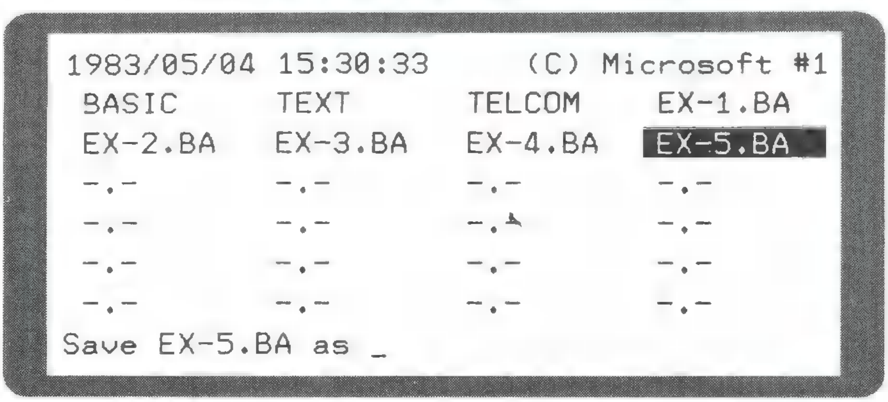
【図174】



【図175】



【図176】



最後に「Ready?」が表示され，[Y]または[⏎]でセーブを開始します。セーブが完了すると通常のメニューへ戻ります（図179）。

カセットレコーダの操作は，BASICモードにおける操作とまったく同じです。

【図177】

```
1983/05/04 15:30:39      (C) Microsoft #1
 BASIC       TEXT        TELCOM      EX-1.BA
 EX-2.BA     EX-3.BA     EX-4.BA     EX-5.BA
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
Save EX-5.BA as cas:MONDAI
```

【図178】

```
1983/05/04 15:30:52      (C) Microsoft #1
 BASIC       TEXT        TELCOM      EX-1.BA
 EX-2.BA     EX-3.BA     EX-4.BA     EX-5.BA
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
B(inary) or A(scii)? _
```

【図179】

```
1983/05/04 15:31:19      (C) Microsoft #1
 BASIC       TEXT        TELCOM      EX-1.BA
 EX-2.BA     EX-3.BA     EX-4.BA     EX-5.BA
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
 -.-         -.-         -.-         -.-
B(inary) or A(scii)? B Ready? _
```



データファイルのロードとセーブ

カセットレコーダを使ってデータファイルを書き込んだり，データファイルを読み込んだりする方法を説明します。

キーボードから入力されたデータや，プログラム内で処理された結果は，RAM上のファイルとして保存することができますが，メモリに余裕がない時は，外部記憶装置，すなわちカセットへ記録しておきます。カセットテープは取りはずしができますから，必要に応じてテープを取り替えることにより，記録できるデータ量に制限がありません。また，テープを別のパソコンにセットして使うこともできます。ただ RAM 上のファイルを扱うのに比べて，スピードが遅いこと，また，一度に一つのファイルしか使えないこと，追加ができないことなどの欠点があります。

まず，カセットテープ上にデータファイルを作成するプログラムから説明します（図180)。

OPEN, CLOSE, PRINT #の各命令は，RAM 上のファイルの作成と同じように機能します。すなわち，OPEN はファイルの使用宣言，CLOSE は終了宣言で，ファイルごとに，処理の最初と最後に必要です。

OPEN の後に書く（“ ”）の中の cas: は，このファイルがカセットテープ上に作成されることを指定するものですから，省略できません。cas: の後がファイル名です。ファイル名は省略できますが，テープ上に名なしのファイルが作成されますから，省略しない方が無難です。

ファイル名の後は使用目的で，カセットの場合は入力用（FOR INPUT）と出力用（FOR OUTPUT）の2つが使えます。

最後に AS#に続けてファイル番号を書きます。カセットにファイルを作成する場合も，データはバッファを経由して出力されます。MAXFILES で指定した範囲内で，ファイル番号，すなわちどのバッファを使うかを割り当てる必要があります。バッファの数は，一度指定されると新たに指定し直されるまで，そのまま残っていますが，プログラムごとに指定した方が確実です。

PRINT #は，RAM 上のファイルへ書き出すのとまったく同じ命令です。OPEN 命令で使用する装置名を指定し，ファイル番号を割り当てれば，その番号が，すべてのファイル関係の命令をカセットテープに対するものとして結びつけるからです。

テープに書き込みを行う場合は，プログラムのセーブの時と同じように，あらかじめカセットレコーダを録音状態にしておくことと，

【図180】

```
10 MAXFILES=1
20 OPEN"cas:testfl" FOR OUTPUT AS#1
30 SCREEN 0,0:CLS:Y=1
40 PRINT "NO      name          mark"
50 LOCATE 0,Y:INPUT A
60 IF A=999 THEN CLOSE:END
70 LOCATE 8,Y:INPUT B$
80 LOCATE 20,Y:INPUT C
90 PRINT #1,A;",";B$;",";C
100 Y=Y+1:IF Y<6 THEN 50 ELSE 30
```

【図181】

```
10 MAXFILES=1
20 OPEN"cas:testfl" FOR INPUT AS#1
30 SCREEN 0,0:CLS:Y=1
40 PRINT "NO      name          mark"
50 IF EOF(1) THEN CLOSE:END
60 INPUT #1,A,B$,C
70 LOCATE 0,Y:PRINT A
80 LOCATE 8,Y:PRINT B$
90 LOCATE 20,Y:PRINT C
100 Y=Y+1:IF Y<6 THEN 50
110 I$=INKEY$:IF I$><CHR$(13) THEN 110
120 GOTO 30
```

テープをカウンタ3位まで進めておくことを忘れないでください。あとは，リモート機能によりテープは動いたり止まったりします。

次に，カセットからデータファイルを読み込むプログラムを説明します（図181）。

OPEN, CLOSE はすでに説明したとおりです。INPUT#は，ファイルの読み込み命令，EOF はファイルの終わりをチェックする関数です。

プログラムを実行する時は，カセットレコーダに必要なデータファイルのテープをセットし，再生状態にしてはじめてください。

カセットテープへデータファイルを書き込んだり，カセットテープからデータファイルを読み込んだりするプログラムは，以上のようにRAM上のテキストファイルを扱うのとまったく変わりません。違いは次の点だけです。

①OPEN の時に使えるのは，INPUT と OUTPUT のモード（APPEND は使えない）。

②ファイル名を指定する場合，装置名の cas: を必ず書く（RAM: の場合は省略）。

③カセットテープ上に複数のファイルを同時に OPEN できない。

⑧ プリンタによる出力

プリンタの接続

PC-8201 のプリンタインターフェイスは，セントロニクス仕様に準拠していますから，PC シリーズのプリンタはいずれも接続できます。

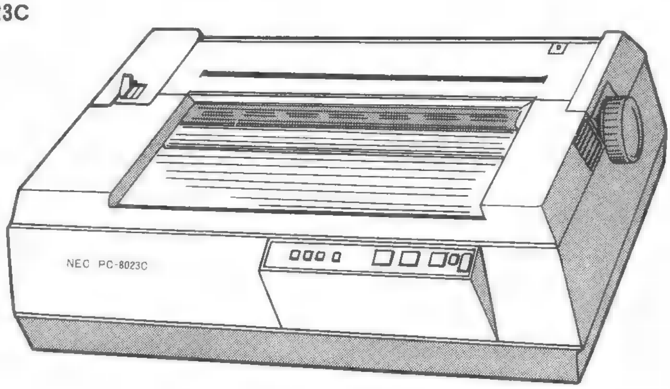
ここでは，PC-8023C（図182）を接続して，プリンタによる出力例を説明します。

まず，PC-8201 と PC-8023C をプリンタ用ケーブル（PC-8294）を使って接続します。接続する時はいつもそうですが，各装置の電源は切ってから行います（図183）。

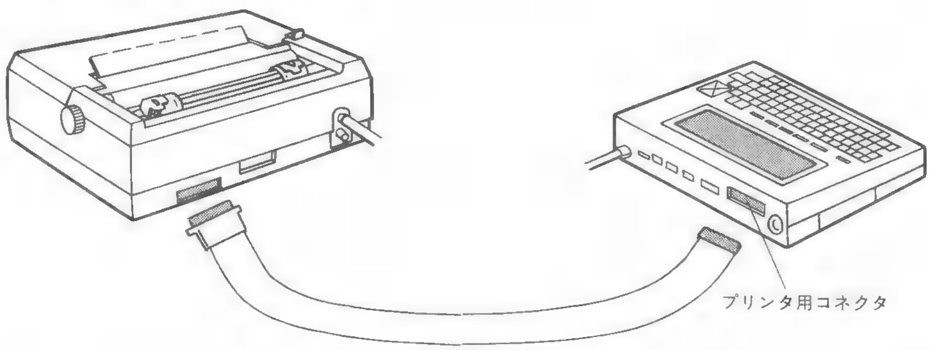
次にインクリボンを取りつける手順を説明します。PC-8023C 用のインクリボンはカートリッジ型なのでセットは簡単です（図184）。

①プリンタの上部後方のペーパーカバーと上部前方のキャリアカバーをはずします（図

【図182】 PC-8023C



【図183】 PC-8023C プリンタとの接続



185)。

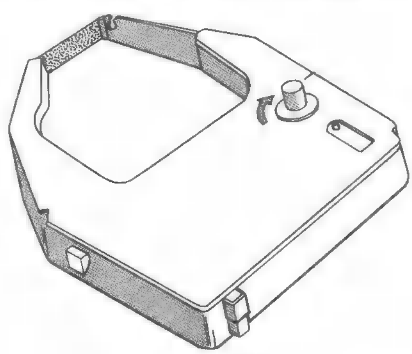
②紙押さえの小さいローラーをじゃまにならない位置へずらし，リボンカートリッジをリボンデッキに取り付けます（図186)。

③インクリボンがリボンガイドとリボンガイドプレートの間に入るようにし，たるみがあればリボンカートリッジについているつまみを右へ回してピンと張ります。

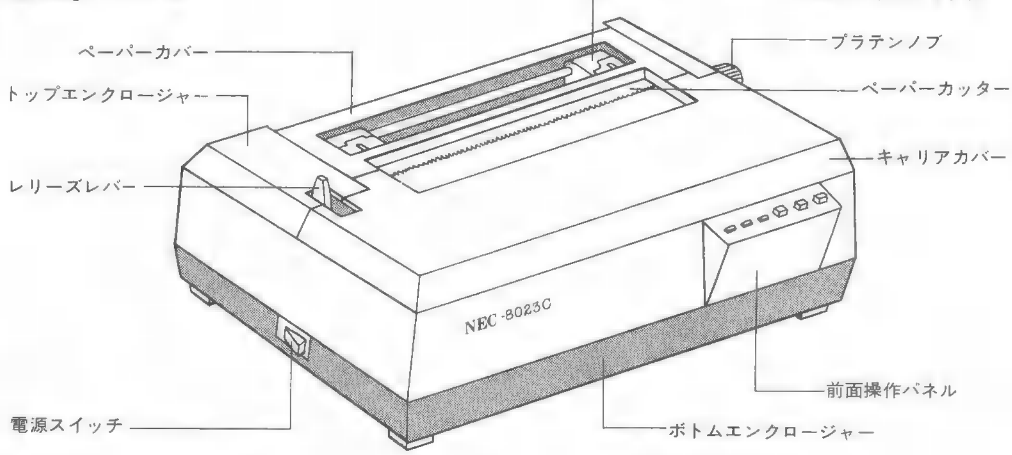
インクリボンの使用期間は頻度によりますが，大体2～3ヵ月位たつと，インクが薄くてみづらくなり，それ以上使い続けているとリボンが破けます。破れる前に交換するようにしてください。

④はずしたカバーを元へ戻します。続けて用紙をセットする時は，それが終わってからカバーを取りつけた方がよいでしょう。

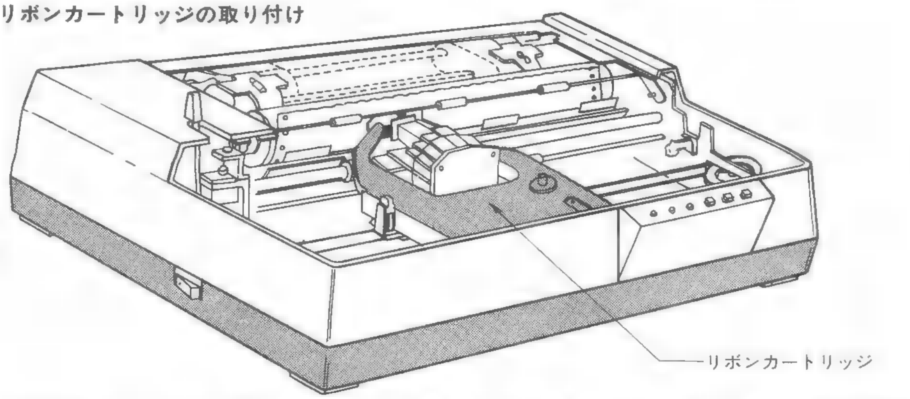
【図184】 リボンカートリッジ



【図185】 プリンタ



【図186】 リボンカートリッジの取り付け



では，用紙のセットを説明します。

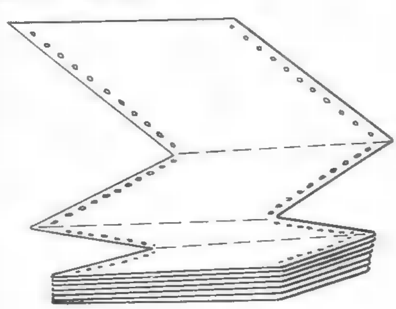
一枚ずつ用紙をセットする場合もありますが，連続的に印刷するために通常ミシン目の入った連続帳票を使います（図187）。

①プリンタ後方のペーパーカバーをはずし，プラスチックのペーパーカッターを前へ倒します。キャリアカバーをはずしている時は必要ありません。

②紙押さえの小さいローラーがついているローラーシャフトを前に引きます(図188)。

③プリンタ左上にあるレリーズレバーを後方に倒して，プラテン（用紙を巻きつける黒

【図187】 連続帳票





いゴムの円柱部分）と紙押さえの間に隙間をつくります。

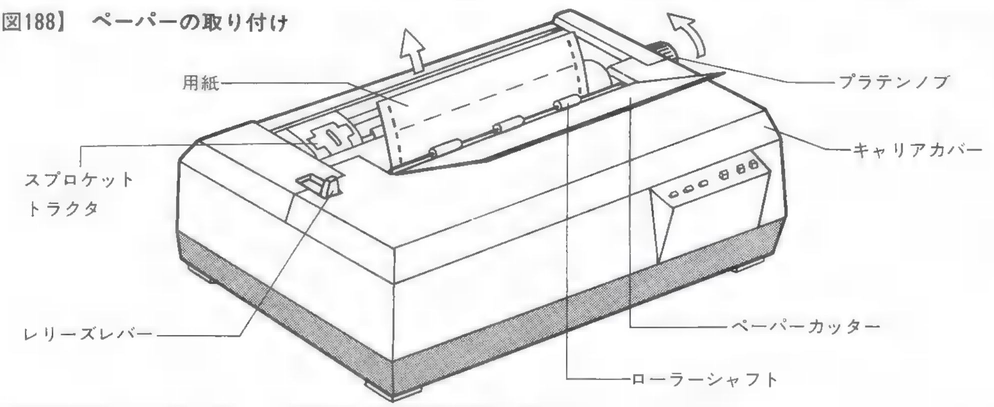
④後方からプラテンに沿って用紙を挿入します。用紙を差し込んだ後，レリーズレバーを前に倒して紙を固定し，プラテンのノブを時計方向へ回すと，簡単に紙が上ってきます。

⑤もう一度，レリーズレバーを後方に倒して紙押さえをゆるめ，用紙の左右にあいている紙送りの穴（スプロケットホール）と，プリンタのスプロケットピンを合わせます（図189）。

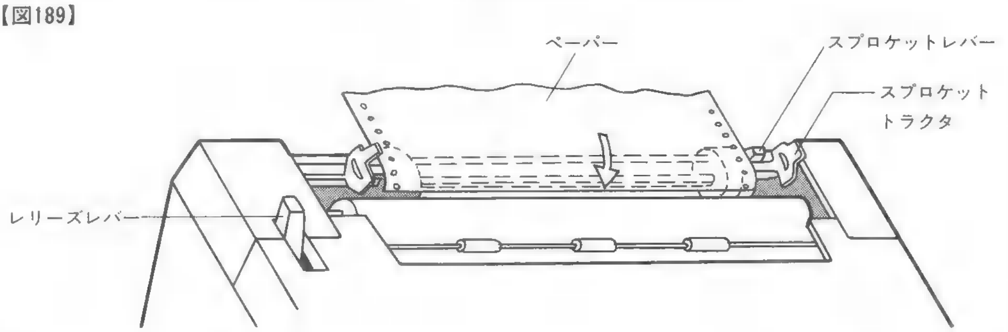
⑥ピンと用紙の穴が合わない時は，スプロケットピンの上にレバーがついているので向う側に倒し，スプロケットトラクター（スプロケットピンのついている部分）の位置を左右に動かして調整できます。

⑦用紙の位置が決ったら，スプロケットレバーを手前に倒してスプロケットトラクターの位置を固定します。それから，スプロケ

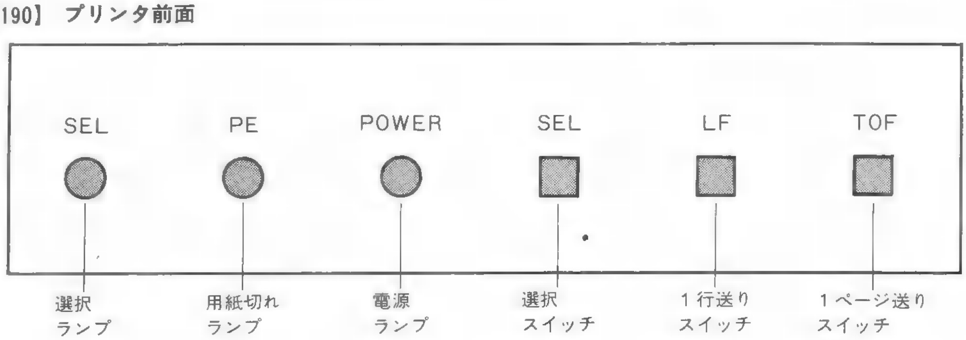
【図188】 ペーパーの取り付け



【図189】



【図190】 プリンタ前面



ットトラクターの紙押さえを倒して紙をピンに固定します。ローラーシャフトを元へ戻し，レリーズレバーを手前に倒して，紙をしっかりプラテンに巻きつけます。

⑧ペーパーカバーを元どおりとりつけます。

言葉でいうと難しそうですが，1，2回実際に用紙をセットしてみれば，すぐ覚えられます。また，用紙は一度セットすれば，なくなるまで（1箱に1000〜2000枚）そのまま自動的に紙送りされます。

インクリボンと用紙のセットが終わったら，いよいよ印刷できます。プリンタの電源スイッチを入れます。プリンタ前面にある3つのランプのうちの右側が緑色になります。ランプのうち右側は電源がONのことを示すPOWERランプ，まん中が用紙切れを示すPEランプ，左側がプリンタが選択中であることを示すSELランプです（図190）。

PEランプは用紙がなくなった時に赤く点灯します。SELランプは，プリンタが選択中（プリント可能状態）の時，緑色に点灯します。プリンタを選択状態にするためには，ランプ右端の SEL スイッチを押します。SELスイッチは1回押すとONに，もう一回押すとOFFになります。

LFとTOFのスイッチは，それぞれ1行，1ページの紙送りをするためのスイッチです。プリンタが選択されていない時，すなわち，SELランプがついていない時のみ働きます。1ページは特に指定しなければ6分の1インチを1行として66行分（11インチ）です。

LFスイッチで行送りした場合と，印刷命令で行送りした場合は，プリンタ自身が行数をカウントし，TOFスイッチが押された時は66－カウント，すなわち残りの行を紙送りしますので，必ずページの最初に用紙が送られます。

最初に紙をセットする時，ミシン目が1行目にくるようにしなかったり，手で用紙を巻き上げたりすると，もちろんTOFスイッチを押しても用紙の最初にはきませんから注意してください。

では，プリンタの電源がONで，SELランプが点灯している状態で次の命令を入力してみましょう。

LPRINT "ABCDEF" [⏎]

LPRINTはプリンタに出力する命令です。ABCDEFが印刷されるはずです。LPRINTの後に続ける定数や変数の規則についてはPRINT命令とまったく同じです。

プリンタによる機能の違い

PCシリーズのプリンタはいずれもプログラマブルなプリンタなので，プリンタ独自の制御コードを送ることにより，印刷方法を変えることができます。制御コードはプリンタの種類により異なりますが，ここではPC-8023Cを例に，その主な機能を紹介します。

印字文字の種類

①ノーマル印字（パイカモード＝通常）

LPRINT CHR$ (27) ; "N";

②12C/Iの印字（エリートモード）

LPRINT CHR$ (27) ; "E";

③17C/Iの印字（縮小モード）

LPRINT CHR$ (27) ; "Q";

④プロポーショナルモード

LPRINT CHR$ (27) ; "P";

カタカナひらがな切り換え

①カタカナ印字（通常）

LPRINT CHR$ (27) ; "$";

②ひらがな印字

LPRINT CHR$ (27) ; "&";

文字の拡大

①拡大文字印字

LPRINT CHR$ (14);

②拡大文字解除

LPRINT CHR$ (15);

改行幅の切り換え

①6行／インチ（通常）

LPRINT CHR$ (27); "A";

②8行／インチ

LPRINT CHR$ (27); "B";

③N行／144インチ

LPRINT CHR$ (27); "T"; "NN";

アンダーライン

①アンダーライン印字

LPRINT CHR$ (27); "X";

②アンダーライン解除

LPRINT CHR$ (27); "Y";

改ページ（TOF スイッチと同じ働き）

LPRINT CHR$ (12)

プリンタの制御コードはLPRINT命令を使ってプリンタへ伝えます。制御コードを受けとると，次に制御コードを受けとるか，プリンタのスイッチを切るまで，同じ印刷状態が続きます（図191）。

図192の例は，アルファベットとカタカナを，制御コードにより印刷方法を変えながら，印刷したものです。

【図191】

```
10 DATA E,ｴﾘｰﾄ,Q,ｼｭｸｼｮｳ,P,ﾌﾟﾛﾎﾟｰｼｮﾅﾙ,N,ﾉｰﾏﾙ
20 DATA &,ﾋﾗｶﾅ,$,ｶﾀｶﾅ,B,8ｷﾞｮｳ,A,6ｷﾞｮｳ
30 DATA X,ｱﾝﾀﾞｰﾗｲﾝ,Y,ｱﾝﾀﾞｰﾗｲﾝｶｲｼﾞｮ
40 FOR I=1 TO 10
50   READ A$,B$
60   LPRINT CHR$(27);A$;
70   IF A$="$" OR A$="A" OR A$="Y" THEN 120
80   LPRINT:LPRINT CHR$(14);" *";B$
90   GOSUB 140
100  LPRINT CHR$(15);
110  GOSUB 140
120 NEXT
130 END
140 LPRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
150 LPRINT "ｱｲｳｴｵ ｶｷｸｹｺ ｻｼｽｾｿ ﾀﾁﾂﾃﾄ"
160 LPRINT "1234567890-^¥!#$%&'()=[];:"
170 RETURN
```



【図192】 制御コードによる印字文字の種類と機能

※エリート

ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:

※シュクショウ

ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:

※プロポーショナル

ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:

※ノーマル

ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:

※ひらかな

ABCDEFGHIJKLMNOPQRSTUVWXYZ
あいうえお　かきくけこ　さしすせそ　たちつてと
1234567890-^¥!#$%&'()=[];:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
あいうえお　かきくけこ　さしすせそ　たちつてと
1234567890-^¥!#$%&'()=[];:

※8ギョウ

ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:

※アンダーライン

ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
アイウエオ　カキクケコ　サシスセソ　タチツテト
1234567890-^¥!#$%&'()=[];:

ファイルの印刷

BASIC モードにある時，BASIC 領域にあるプログラムを印刷することができます。画面にプログラムを表示する LIST に似た LLIST（エルリスト）の命令を使います。

まず BASIC 領域内にプログラムを入れます。簡単なプログラムを入力するか，RAM 上のプログラムをロードしてください。次に [L] [L] [I] [S] [T] [⏎] と入力します。画面に表示するのが印刷に変わっただけでプログラムのリストが出力されます。LLIST は，LIST と同じように行番号を指定して印刷することもできます。

①LLIST　行番号　その行動番号のみ印刷
②LLIST　行番号—行番号　範囲内を印刷
③LLIST　行番号—　その行番号から最後まで印刷
④LLIST　—行番号　最初からその行番号まで印刷

LLIST が終わっても BASIC モードのままです。LLIST を途中でやめたい時は，[SHIFT] キーを押しながら [STOP] キーを押してください。印刷中のプリンタは，[STOP] だけでは止まりません。

メニューモードにある場合は，RAM 上のファイルを簡単に印刷することができます。メニューモードでの印刷には，[f·4] に登録されている LIST コマンドを使います。

メニューモードにおける LIST コマンドは，BASIC ファイル（.BA），TEXT ファイル（.DO）の両方を印刷する機能を持っています。

まず，印刷したいファイルの名前にカーソルを移動してから [f·4] を押します。印刷しようとしているファイルがテキストファイルの場合は，続いて画面の一番下の行に，印字幅についての質問が出ます（図193）。

印字幅とは，印刷する時に1行に最大何文字印刷するかのことで，10～132で指定します。指定を省略して [⏎] のみ入力した場合は，( )内の数字を指定したものとみなされます。( )内の数字は，印字幅を指定するたびに書き替わります。電源を切っても，この省略値は消えません。

BASIC ファイルを印刷する場合は，印字幅は指定できません。プリンタが可能な印字幅で印刷します。すなわち，PC-8023C では，特に拡大や縮小を指定していなければ1行に80字印刷します。印字幅の指定ができないので，TEXT ファイルにカーソルを重ねて，[f·4] を押しても，印字幅の質問は出ません。

RAM 上で LIST を指定すると，最後に

【図193】

```
1983/05/06 11:24:27      (C) Microsoft #1
 BASIC      TEXT       TELCOM     telF.DO
 EX-1.BA    EX-2.BA    EX-3.BA    -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
List Width(80): _
```

【図194】

```
1983/05/06 11:24:29     (C) Microsoft #1
 BASIC     TEXT      TELCOM    telF.DO
 EX-1.BA   EX-2.BA   EX-3.BA   -.-
 -.-       -.-       -.-       -.-
 -.-       -.-       -.-       -.-
 -.-       -.-       -.-       -.-
 -.-       -.-       -.-       -.-
List telF.DO Ready? _
```

プリンタの準備ができているかをきいてきます（図194）。

［Y］または［⏎］を入力すると，印刷を開始します。［Y］以外を入力すると，［f･4］を押す前の状態に戻ります。

また，［Y］あるいは［⏎］キーで印刷を開始した時に，実際にはプリンタの準備ができていなくて印刷ができないことがあります。この場合特にエラーの表示はされません。PC-8201は，そこで止まってしまいます。プリンタの準備をしてセレクト状態にすると印刷を行います。一旦印刷状態から抜けたい場合は，［SHIFT］キーを押しながら［STOP］キーを押します。

印刷プログラムの作成

すでに述べたように，プリンタに出力するための命令はLPRINTです。LPRINTは，出力先がプリンタであること以外は，画面表示のPRINTとほぼ同じ働きをします。

図195のプログラムを実行してみましょう。またPRINT命令をLPRINT命令に修正した後，もう一度実行してその違いをみてください（図196）。

PRINTとLPRINTの命令に差はなくても，印刷のプログラムを作るためには，画面表示させる場合とは違った配慮が必要になります。なぜならば，画面はいま操作しながらみているわけですが，印刷したものは，操作とは関係のない人が，関係のない時点にみるかもしれません。そのため，いつ，何のために印刷したものか，つまり日付や標題が印刷



の中に必要となります。

また，画面は横40字，縦8行の枠の中でみやすい形式を考えるわけですが，プリンタでは標準の用紙に，標準サイズの文字で，横80字，縦66行の印刷ができます。さらに，プリンタの機能によっては，文字の縮小や拡大を行うこともできます。また，連続した用紙は1ページごとにページ送りも必要です。

部，日付，コード，数量からなる，SYOHINを読んで印刷する例をあげます。見出し，ページ，小計，合計，総合計の計算，ページ送りなどを参考にしてください。ただし，データは部，日付の順に並んでいるのが条件です（図197）。

【図195】

```
10 SCREEN 0,0:CLS
20 PRINT TAB(4)"**  TELEPHONE  **"
30 PRINT "NAME" TAB(20) "TEL-NO"
40 FOR I=1 TO 8
50 READ A$,B$
60 PRINT A$ TAB(20) B$
70 NEXT
80 END
90 DATA hirooka,499-2671,tabuchi,584-4955
100 DATA anabuki,233-4566,kagawa,662-8752
110 DATA ando,674-3244,kobayashi,882-6759
120 DATA oosawa,745-3240,enatu,473-1040
```

【図196】

```
    **  TELEPHONE  **
NAME                TEL-NO
hirooka             499-2671
tabuchi             584-4955
anabuki             233-4566
kagawa              662-8752
ando                674-3244
kobayashi           882-6759
oosawa              745-3240
enatu               473-1040
```



【図197-①】

```
10 ' ***************************
20 ' *                         *
30 ' *     ｲﾝｻﾂ ﾌﾟﾛｸﾞﾗﾑ ｻﾝﾌﾟﾙ     *
40 ' *                         *
50 ' ***************************
60 GOSUB 1230
70 OPEN "SYOHIN" FOR INPUT AS #1
80 IF EOF(1) THEN 170
90 INPUT #1,A$,D$,G$,S,T
100 IF SW= 0 THEN SW=1 :K1$=A$:K2$=D$
110 IF K1$<>A$ THEN GOSUB 1090 :GOSUB 1140:GOSUB 1230:GOTO 140
120 IF K2$<>D$ THEN GOSUB 1090
130 IF GYO>=50 THEN GOSUB 1230
140 GOSUB 1030
150 SG#=SG#+K#
160 GOTO 80
170 'END
180 GOSUB 1090:GOSUB 1140:GOSUB 1180
190 CLOSE:END
1000 '******************
1010 '*  sub routine    *
1020 '******************
1030 'print routine
1040 LPRINT TAB(5) A$ TAB(10) D$ TAB(21) G$;
1050 LPRINT TAB(32) USING"######,";S;
1060 LPRINT TAB(45) USING"######,";T;
1065 K#=S*T
1070 LPRINT TAB(60)USING"##########,";K#
1080 GYO=GYO+1:RETURN
1090 ' sub total
1100 LPRINT TAB(5)"sub total";
1110 LPRINT TAB(60) USING"##########,";SG#
1120 LPRINT STRING$(80,"-"):GYO=GYO+2
1130 MG#=MG#+SG#:SG#=0:K2$=D$:RETURN
1140 ' total
1150 LPRINT TAB(5) "total";
1160 LPRINT TAB(60) USING"##########,";MG#
1170 G#=G#+MG#:MG#=0:K1$=A$:K2$=D$:RETURN
1180 'grand total
1190 LPRINT
1200 LPRINT TAB(5) "grand total";
1210 LPRINT TAB(60) USING"##########,";G#
1220 RETURN
1230 ' TITLE PRINT
1240 IF SW=1 THEN LPRINT CHR$(12)
1245 LPRINT CHR$(14);
1250 LPRINT TAB(12);CHR$(27);"X";"ｳﾘｱｹﾞ ｲﾁﾗﾝ ﾋﾟｮｳ"
1260 LPRINT CHR$(15):LPRINT CHR$(27);"Y";
1263 LPRINT TAB(5) DATE$;
1265 P=P+1:LPRINT TAB(70)USING "page ##";P:LPRINT
1270 LPRINT TAB(5) "ﾌﾞ" TAB(10) "ﾋﾂﾞｹ";
1280 LPRINT TAB(20) "ｺｰﾄﾞ" TAB(30) "ｽ ｳ ﾘ ｮ ｳ";
1290 LPRINT TAB(45) "ﾀ  ﾝ  ｶ" TAB(63) "ｷ ﾝ ｶﾞ ｸ"
1300 LPRINT :GYO=0:RETURN
```

【図197-②】

<u>ｳﾘｱｹﾞ ｲﾁﾗﾝ ﾋﾟｮｳ</u>

58/03/31 page 1

ﾌﾞ	ﾋﾂﾞｹ	ｺｰﾄﾞ	ｽｳﾘｮｳ	ﾀﾝｶ	ｷﾝｶﾞｸ
01	580301	01	500	4,000	2,000,000
01	580301	02	280	45,000	12,600,000
01	580301	03	320	8,300	2,656,000
01	580301	04	800	12,000	9,600,000
01	580301	05	400	11,000	4,400,000
sub total					31,256,000
01	580305	01	1,500	40,000	60,000,000
01	580305	02	190	7,700	1,463,000
01	580305	03	1,000	11,000	11,000,000
01	580305	04	900	5,500	4,950,000
01	580305	05	350	12,000	4,200,000
01	580305	06	100	3,200	320,000
sub total					81,933,000
01	580307	01	1,500	4,000	6,000,000
01	580307	15	190	14,000	2,660,000
01	580307	23	100	51,000	5,100,000
01	580307	10	900	18,000	16,200,000
01	580307	07	350	12,300	4,305,000
01	580307	06	100	3,200	320,000
sub total					34,585,000
01	580308	03	1,000	11,000	11,000,000
01	580308	05	500	91,000	45,500,000
01	580308	10	1,000	15,000	15,000,000
01	580308	04	900	1,500	1,350,000
01	580308	05	350	15,000	5,250,000
01	580308	06	100	2,500	250,000
01	580308	07	1,000	6,200	6,200,000
01	580308	10	500	3,200	1,600,000
sub total					86,150,000
01	580309	01	1,000	11,000	11,000,000
01	580309	02	920	20,000	18,400,000
01	580309	03	740	32,000	23,680,000
01	580309	11	400	28,500	11,400,000
sub total					64,480,000
01	580311	01	1,500	2,500	3,750,000
01	580311	02	190	6,300	1,197,000
01	580311	09	350	2,000	700,000
01	580311	06	100	3,200	320,000
sub total					5,967,000
total					304,371,000

【図197-③】

ウリアゲ゛ イチラン ヒ゜ョウ

58/03/31 page 2

ブ゛	ヒヅ゛ケ	コード゛	スウリョウ	タンカ	キンガ゛ク
02	580301	01	700	4,000	2,800,000
02	580301	02	300	8,300	2,490,000
02	580301	03	1,000	11,000	11,000,000
02	580301	04	650	5,500	3,575,000
sub total					19,865,000
02	580303	02	450	8,300	3,735,000
02	580303	06	1,000	3,200	3,200,000
sub total					6,935,000
02	580305	01	1,500	4,000	6,000,000
02	580305	04	190	4,000	760,000
02	580305	13	1,000	4,300	4,300,000
02	580305	40	560	5,500	3,080,000
02	580305	05	350	12,000	4,200,000
02	580305	11	100	3,200	320,000
sub total					18,660,000
02	580307	01	300	4,800	1,440,000
02	580307	51	1,500	4,000	6,000,000
02	580307	41	700	4,000	2,800,000
02	580307	42	190	8,300	1,577,000
02	580307	02	300	8,300	2,490,000
02	580307	03	1,000	1,000	1,000,000
02	580307	04	600	5,500	3,300,000
sub total					18,607,000
total					64,067,000

【図197-④】

ｳﾘｱｹﾞ ｲﾁﾗﾝ ﾋﾟｮｳ

58/03/31 page 3

ﾌﾞ	ﾋﾂﾞｹ	ｺｰﾄﾞ	ｽｳﾘｮｳ	ﾀﾝｶ	ｷﾝｶﾞｸ
03	580301	05	350	12,000	4,200,000
03	580301	17	1,000	12,000	12,000,000
03	580301	06	1,000	3,200	3,200,000
03	580301	07	1,500	3,200	4,800,000
sub total					24,200,000
03	580302	02	190	8,700	1,653,000
03	580302	03	1,000	16,000	16,000,000
03	580302	04	500	2,900	1,450,000
sub total					19,103,000
03	580306	05	190	21,000	3,990,000
03	580306	06	100	4,000	400,000
sub total					4,390,000
03	580307	01	1,500	4,110	6,165,000
03	580307	02	190	8,000	1,520,000
03	580307	03	7,500	11,000	82,500,000
03	580307	04	200	5,500	1,100,000
03	580307	05	400	12,000	4,800,000
03	580307	06	100	3,200	320,000
sub total					96,405,000
03	580308	01	330	4,000	1,320,000
03	580308	02	100	8,300	830,000
03	580308	03	560	11,000	6,160,000
03	580308	04	800	5,500	4,400,000
03	580308	05	400	12,000	4,800,000
03	580308	06	1,000	3,200	3,200,000
sub total					20,710,000
03	580309	11	500	14,000	7,000,000
03	580309	12	90	3,600	324,000
03	580309	13	900	13,100	11,790,000
03	580309	14	900	30,500	27,450,000
03	580309	15	950	6,600	6,270,000
03	580309	16	300	3,300	990,000
sub total					53,824,000
total					218,632,000
grand total					587,070,000

⑨ N82-BASIC補足

ファンクションキー

キーボード最上段についている f·1 ～ f·5 のファンクションキーの働きについては，第1章のキー操作でふれましたが，BASIC モードにおける各ファンクションキーの内容について説明しましょう。

ファンクションキーの内容は，工場出荷時およびコールドスタート時には**表7**のようになっています。

f·1 に登録されている「Load "」は，すでに何度も使ってきたプログラムのロードを行う命令です。f·1 を押して LOAD " を表示した後，プログラム名を入力します。

f·2 に登録されている「Save "」は，プログラムのセーブを行う命令です。f·2 を押した後，プログラム名を入力します。

f·3 に登録されている「Files」は，現在 RAM 上にあるすべてのファイルの名前を表示します。FILES の後に装置番号をつけると，RAM 以外の装置（0の時 SIO インターフェイスに接続されている外部記憶装置，1～4の時フロッピーディスク装置）上のファイル名を表示します。

BASIC, TEXT, TELCOM の各ソフトウェアは RAM 上にありませんから表示されません。表示されるのは，現在選択されているバンク上のファイルだけです。f·3 には，FILES の文字のほかに，⏎ キーの機能も合わせて登録されているため，⏎ は押す必要がありません（**図198**）。

f·4 に登録されている LIST は，現在 BASIC 領域にあるプログラムを表示する命令です。f·4 を押した後 ⏎ キーを押すとプログラムの小さい行番号から表示されます。LIST に続けて行番号を書くと，特定の行だけを表示できます。

①LIST 行番号 ⏎　その行のみ
②LIST 行番号—行番号 ⏎　範囲を指定
③LIST—行番号 ⏎　最初～指定行まで

【表7】 BASICモードにおけるファンクションキー

キー番号	キー	内容	機能
1	f·1	LOAD"	プログラムをBASIC領域に読み込む
2	f·2	SAVE"	BASIC領域のプログラムを他へ書き出す
3	f·3	FILES	RAM上のファイルの表示
4	f·4	LIST	プログラムの表示
5	f·5	RUN	プログラムの実行
6	SHIFT + f·1	EDIT	TEXTソフトウェアへ移る
7	SHIFT + f·2	CONT	中断したプログラムの再開
8	SHIFT + f·3	PRINT	画面表示
9	SHIFT + f·4	LIST.	BASICが注目している行を表示
10	SHIFT + f·5	MENU	メニューモードへ戻る

【図198】

```
Files
EX-1  .BA   EX-2  .BA  EX-30a.BA*
Ok
■




Load "  Save "  Files  List    Run
```

④ LIST 行番号— [⏎]　指定行〜最後まで

[f·5] に登録されている「Run」というのは，BASIC 領域にあるプログラムを実行する命令です。

[f·5] にはRUNのほかに，[⏎] キーの機能も合わせて登録されているため，[⏎] を押す必要はありません。RUN 命令には，特定の行から実行を開始するとか，RAM上のファイルから読み込んで実行するなどの機能もありますが，[f·5] を押した場合は，必ず BASIC 上のプログラムを一番小さい行から実行することになります。

[f·6] に登録されている「Edit」は，BASIC モードから TEXT モードに移る働きをします。実際に試すために，図199のプログラムを入力してください。

入力が終わったら，[f·6] を押します。EDIT の文字が表示されます。そのまま [⏎] キーを押すとプログラム全体が，行範囲を指定するとプログラムのその部分が TEXT で編集できます。

ここでは，EDIT に続いて，20—50を入力します（図200）。

[⏎] を押すと，TEXT モードに入り，画面に指定の範囲が表示されます。画面に全部表示できなくても，スクロールはしません（図201）。

現在表示されているのはプログラムの一部ですが，TEXT モードでは，単なる文字列と考えます。修正はカーソルを移動して行いますが，修正ごとに [⏎] キーを押す必要はありません。画面上の文字列が，実際の

【図199】

```
10 SCREEN 0,0:CLS
20 FOR I= 1 TO 3
30 SOUND 2345,4:SOUND 3844,4:SOUND 2956,4
40 NEXT
50 A=INT(RND(1)*26)
60 A$=CHR$(65+A)
70 INPUT "ｱﾙﾌｧﾍﾞｯﾄ ｦ ｱﾃﾃ ｸﾀﾞｻｲ";X$
80 IF A$=X$ THEN PRINT "ｵｵｱﾀﾘ !!":GOTO 110
90 C=C+1:IF C<10 THEN GOTO 70
100 PRINT "ｻﾞﾝﾈﾝﾃﾞｼﾀ"
110 PRINT " PRESS ANY KEY"
120 B$=INKEY$
130 IF B$="" THEN GOTO 120
140 C=0:GOTO 10
```

【図200】

```
90 C=C+1:IF C<10 THEN GOTO 70
100 PRINT "ｻﾞﾝﾈﾝﾃﾞｼﾀ"
110 PRINT "PRESS ANY KEY"
120 B$=INKEY$
130 IF B$="" THEN GOTO 120
140 C=0:GOTO 10
Edit 20-50
Load "  Save "  Files  List    Run
```

【図201】

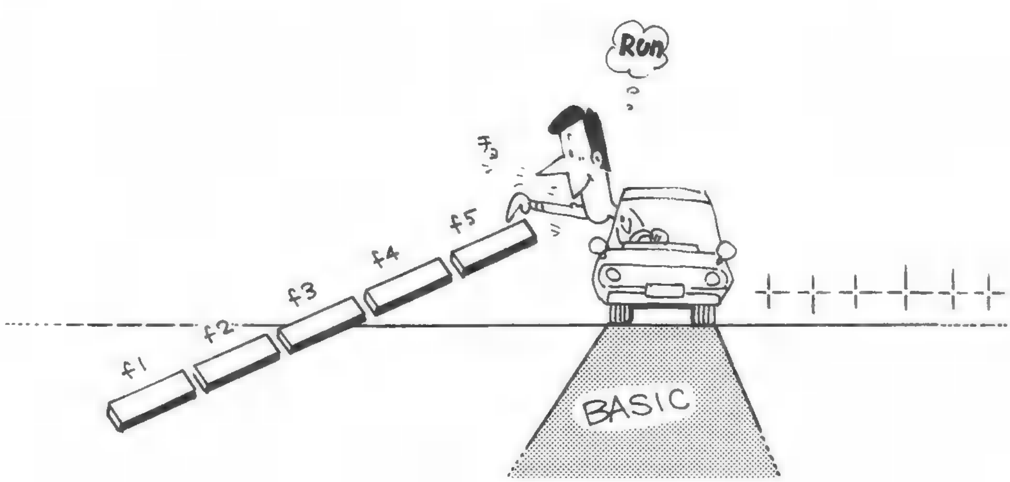
```
20 FOR I=1 TO 3⏎
30 SOUND 2345,4:SOUND 3844,4:SOUND 2956,
4⏎
40 NEXT⏎
50 A=INT(RND(1)*10)⏎
◀
```

RAM 上に記憶されます。どこか1文字修正してください。

修正が終わったら，f·10 を押すか，または ESC キーを2回押します。すると，もとの BASIC モードへ戻ります。LIST をとって，プログラムの修正が行われているのを確認してください。

EDIT を使って，BASIC モードから TEXT モードに入りプログラムを修正すると，TEXT の持つ機能を有効に使えるため，BASIC モードで画面に表示されるプログラムを修正しながら ⏎ キーを押すよりもずっと便利です。TEXT の機能については第3章の TEXT を参照してください。

f·7 に登録されている「Cont」は，STOP により中断されたプログラムを再開し



【図202】

```
ｱﾙﾌｧﾍﾞｯﾄ ｦ ｱﾃﾃ ｸﾀﾞｻｲ?
Break in 70
Ok
■



Load "  Save "  Files  List    Run
```

【図203】

```
ｱﾙﾌｧﾍﾞｯﾄ ｦ ｱﾃﾃ ｸﾀﾞｻｲ?
Break in 70
Ok
A$="P"
Ok
Cont
ｱﾙﾌｧﾍﾞｯﾄ ｦ ｱﾃﾃ ｸﾀﾞｻｲ? ■
Load "  Save "  Files  List    Run
```

ます。[STOP]キー，STOP 命令，[CTRL]キーと[C]のいずれかを入力するとプログラムは，「Break in 行番号」を表示して中断されます（図202）。

[f·7]を押すと再開されます。中断から再開までの間に LIST や変数の表示，変数の値の変更が行われてもかまいませんが，プログラムの修正があった場合は，CONT は働きません（図203）。

[f·7]には「Cont」のほかに[⏎]キーの機能も登録されているため[⏎]は必要ありません。

[f·8]に登録されている「Print」は，もちろん画面表示のための命令です。

[f·9]に登録されている「List.」は，現在 BASIC が着目している行を画面表示させる働きをします。ピリオドは LIST 以外でも使うことができ，最後に修正した行，エラーを起こした行などの行をさすポインタです。

[f·9]には LIST. のほか[⏎]キーの機能が合わせて登録されていますから，[⏎]を押す必要はありません。

[f·10]に登録されている「Menu」は，メニューモードへ戻る命令です。[⏎]キーの機能も合わせて登録されています。

ファンクションキーの登録は，BASIC モードにある時に限り，使用者が自由に変更することができます。ファンクションキーの登録には KEY 命令を使います（図204）。

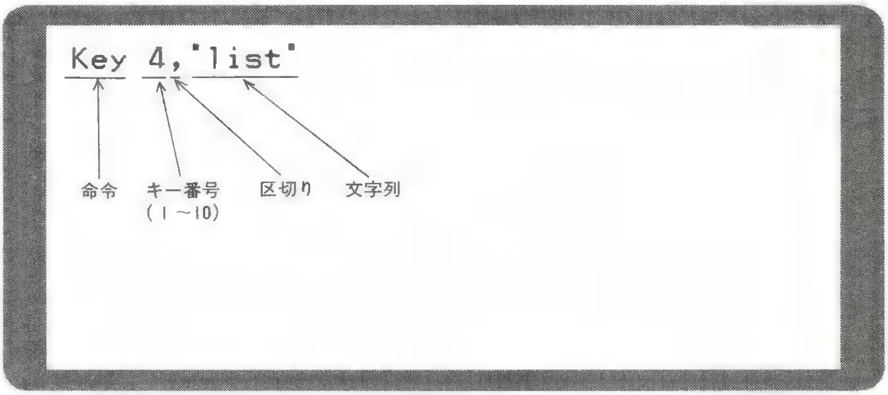
各キーに登録できる文字列の長さは最大15文字までです。キーボードから入力できない文字や機能は CHR$ 関数で入力します。例えば[⏎]キー，ダブルクォーテーション（“）などです。次の例をみてください。

KEY 1, “LOAD”+CHR$（34）

KEY 5, “RUN”+CHR$（13）

ファンクションキーに登録された内容は電源を切っても消えません。

【図204】



デバッグの方法

プログラムが正しく目的どおり機能するかどうかをチェックすることをデバッグ（DEBUG）といいます。

プログラムの誤りには，BASIC の文法どおりに書かれていない場合と，プログラムの目的と実行した状況が違っている場合，つまり論理的におかしい場合とがあります。

PRINT を PLINT とタイプしたとか，INPUT で変数を指定していないとかの場合は，その命令にぶつかった時に，？SN ERROR（文法エラー）と表示されてプログラムが中断されます。このように命令そのものがまちがっている場合は，まちがった命令でプログラムが止まるため比較的簡単にミスが発見できます（図205）。

命令そのものは正しくタイプされていても

【図205】
文法エラー

```
Run
?SN Error in 40
Ok
■



Load "  Save "  Files  List    Run
```

【図206】
行番号エラー

```
Run
?UL Error in 30
Ok
■



Load "  Save "  Files  List    Run
```



エラーになる場合があります。GOTO 5 の命令で，プログラム上に行番号5がなければエラーになります（図206）。

もっと困るのは，実行の条件によって，正しく働く時も，エラーになる時もある場合です。例えば入力されたAの値に基づいて，ON A GOTO 10, 20, 30 の命令を実行する場合，Aの値が1，2，3のいずれかであれば，それぞれ10，20，30へGOTOしますが，3以上ならば命令は無視され，負の値ならば？FC ERRORが発生します。Aの値が負にならないように，また負の値になりうる場合は，ON GOTO の命令をジャンプするようにプログラムを作りかえる必要があります。

エラーによる中断は起きずに，プログラムが最後まで進んでも，正しいとは限りません。計算結果が正しくない，希望の様式になっていないなどの場合です。デバッグでは，考えられるあらゆる条件を想定してテストする必要があります。

N_{82}-BASICでは，デバッグを手助けするためのいくつかの命令が用意されています。

ON ERROR GOTO（オンエラーゴーツー）命令

ON ERROR GOTOは，エラーが発生した場合，分岐するエラー処理ルーチンの行番号を定義する命令です。ON ERROR GOTOは，書かれた行で実行されるのではなく，飛び先の宣言をするだけですから，プログラムの最初に書いておけばどのエラーに対しても共通に働きます。エラー処理ルーチンが宣言されていると，エラーが発生した場合に中断せずに，指定の行へ分岐します。エラー処理ルーチンから元のプログラムへ戻すためにはRESUME（レジューム）命令を使います。エラー処理ルーチン内でさらにエラーが発生した場合は，エラーの内容が表示されて，プログラムは中断されます。

RESUME 命令

RESUMEは，エラー回復処理の後，プログラムの実行を再開します。

RESUMEまたはRESUME 0はエラーが起こった行から再開。

RESUME NEXTはエラーの命令の次から再開。

RESUME 行番号は指定の行番号から再開。

ERL 関数

エラーが発生した行番号を調べる働きをします。ERL は BASIC の予約変数なので使用者が値を代入することはできません。

ERR 関数

発生したエラーの種類を0～255までのコードで与える働きをします。ERL と同様に，BASIC の予約変数なので値を代入することはできません。

ERROR 命令

エラー発生のシュミレーションを行う命令です。ERROR の後に0～255の範囲の整数でエラーコードを指定します。そのコードが BASIC ですでに役割の決まっているものならば（1—25，50—67の時），エラーの内容が表示されます。BASIC で未定義のエラーコードならば，使用者が指定するエラーコードになります。

図207のプログラムを実行し，処理番号に，－1を入力してください。本来は，エラーでプログラムが中断するはずですが，ユーザーが指定したメッセージを表示して続けられます。同じく，処理番号に1，分母に0を入力してください。同様に処理が続けられます（図208）。

【図207】

```
10 ON ERROR GOTO 130
20 SCREEN 0,0:CLS
30 PRINT "1... ｹｲｻﾝ"
40 PRINT "2... ｵﾜﾘ"
50 INPUT "ｼｮﾘ ﾊﾞﾝｺﾞｳ";A
60 ON A GOTO 90,80
70 GOTO 20
80 END
90 PRINT TAB(6) "ｹｲｻﾝ ROUTINE"
100 INPUT "ﾌﾞﾝﾎﾞ ﾆｭｳﾘｮｸ";B:C=A/B
110 PRINT "A/B=";C
120 GOTO 20
130 IF ERL=60 THEN GOTO 160
140 IF ERL=100 THEN GOTO 180
150 PRINT "ｼｮﾘ ﾁｭｳｼ":ERROR 5
160 PRINT "ﾊﾞﾝｺﾞｳ ﾐｽ"
170 RESUME 50
180 PRINT "ﾌﾞﾝﾎﾞ ｶﾞ 0"
190 RESUME 100
```

【図208】

```
2...ｵﾜﾘ
ｼｮﾘ ﾊﾞﾝｺﾞｳ? -1
ﾊﾞﾝｺﾞｳ ﾐｽ
ｼｮﾘ ﾊﾞﾝｺﾞｳ? 1
      ｹｲｻﾝ ROUTINE
ﾌﾞﾝﾎﾞ ﾆｭｳﾘｮｸ? 0
ﾌﾞﾝﾎﾞ ｶﾞ 0
ﾌﾞﾝﾎﾞ ﾆｭｳﾘｮｸ? ■
```

サウンド機能

PC-8201は，N82-BASICに含まれるSOUND（サウンド）命令を使って，5オクターブのさまざまな音を出すことができます。次の命令を実行してください。

SOUND 4697, 32

2オクターブ目のド（ハ長調）の音が，4分音符の長さ演奏されます。同じくレの音は，

SOUND 4184, 32

図209のプログラムを実行してください。ドレミファソラシドが演奏されます。

SOUND命令では，SOUNDの後に音程と長さを数字で指定します。音程には0～16383の範囲の数字を使い，数字が大きい程，低い音になります。長さは0～255の範囲内の数字を使い，1が0.02秒の長さです。

表8からもわかりますが，各音の間，例えばド（4697）とレ（4184）の間に513の音程をとることができます。そのため，音楽的な半音階だけでなく，さまざまな擬音を出すことができます（図210）。

音階表の各音の数字は，元の音の数字を，1.0594639で割って求めています。

【図209】
```
10 FOR I= 1 TO 15:READ A:SOUND A,16:NEXT
20 DATA 9394,8368,7456,7032,6269,5586,4968
30 DATA 4697,4184,3728,3516,3134,2793,2486,2348
```

【表8】 音 階 表

コード＼オクターブ	1	2	3	4	5	6
C		9394	4697	2348	1171	587
C♯		8866	4433	2216	1103	554
D		8368	4184	2092	1045	523
D♯	15800	7900	3950	1975	987	493
E	14912	7456	3728	1864	932	466
F	14064	7032	3516	1758	879	439
F♯	13284	6642	3321	1660	830	415
G	12538	6269	3134	1567	783	
G♯	11836	5918	2954	1479	733	
A	11172	5586	2793	1396	693	
A♯	10544	5272	2636	1316	653	
B	9952	4968	2486	1244	622	

【図210】
```
10 FOR I=1 TO 2:FOR J= 1 TO 4
20 READ A,B:SOUND A,4:SOUND B,6
30 NEXT J:RESTORE
40 NEXT I:SOUND 9086,16
50 DATA 9048,9295,9411,9368
60 DATA 9234,9693,9286,9448
```

キャラクタジェネレータ機能

N_{82}-BASICでは，256種類の文字を扱うことができます。256種類それぞれを区別するために，0～255までのキャラクタコード（文字につけた番号）が決められていて，それをまとめたものをキャラクタコード表と呼びます。キャラクタコード表の決め方もいろいろあるのですが，N_{82}-BASICでは，アスキーコードに準拠しています（表9）。

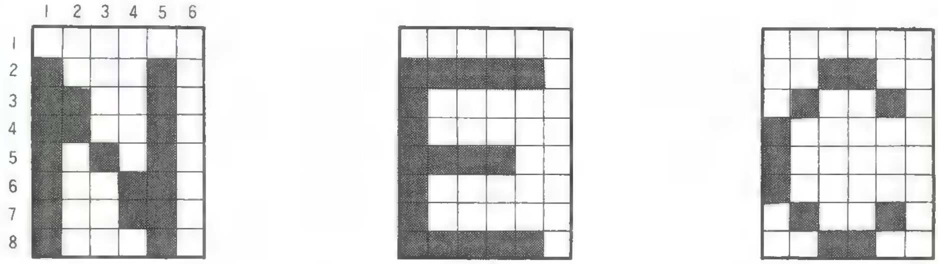
N_{82}-BASICのキャラクタコード表では，その大部分に，数字，アルファベット（大文字と小文字），カタカナ，記号，および文字として表示できない特殊コードが割り当てられていますが，一部未定義のキャラクタコードがあります。これら未定義のキャラクタコードの131～159，224～255の合計61個分に使用者が文字を割り当てることができます。使用する文字は，すでに存在しているアルファベットや数字でもよいのですが，使用者が自由に新しい文字を作って（GENERATEして），キャラクタコード表の中に組み込むことができます。

他のパソコンでは，ゲーム用などの目的でキャラクタコードの空いているところに，グラフィックキャラクタを定義し，そのキャラクタは，GRPH キーとキーボード上の別のキーを一緒に押すと出力できるようにしてあり

【表9】 キャラクタコード表

0		32		64	@	96	`	128	◀	160		192	タ	224	
1		33	!	65	A	97	a	129	↵	161	。	193	チ	225	
2		34	"	66	B	98	b	130	▩	162	「	194	ツ	226	
3		35	#	67	C	99	c	131		163	」	195	テ	227	
4		36	$	68	D	100	d	132		164	、	196	ト	228	
5		37	%	69	E	101	e	133		165	・	197	ナ	229	
6		38	&	70	F	102	f	134		166	ヲ	198	ニ	230	
7	コントロールコード表参照（文字として出力されない特殊コード）	39	'	71	G	103	g	135		167	ァ	199	ヌ	231	
8		40	(	72	H	104	h	136	（ユーザー定義キャラクタ（キーボードから入力可能））	168	ィ	200	ネ	232	（ユーザー定義キャラクタ（CHR$関数を使って出力する））
9		41	)	73	I	105	i	137		169	ゥ	201	ノ	233	
10		42	*	74	J	106	j	138		170	ェ	202	ハ	234	
11		43	+	75	K	107	k	139		171	ォ	203	ヒ	235	
12		44	,	76	L	108	l	140		172	ャ	204	フ	236	
13		45	−	77	M	109	m	141		173	ュ	205	ヘ	237	
14		46	.	78	N	110	n	142		174	ョ	206	ホ	238	
15		47	／	79	O	111	o	143		175	ッ	207	マ	239	
16		48	0	80	P	112	p	144		176	ー	208	ミ	240	
17		49	1	81	Q	113	q	145		177	ア	209	ム	241	
18		50	2	82	R	114	r	146		178	イ	210	メ	242	
19		51	3	83	S	115	s	147		179	ウ	211	モ	243	
20		52	4	84	T	116	t	148		180	エ	212	ヤ	244	
21		53	5	85	U	117	u	149		181	オ	213	ユ	245	
22		54	6	86	V	118	v	150		182	カ	214	ヨ	246	
23		55	7	87	W	119	w	151		183	キ	215	ラ	247	
24		56	8	88	X	120	x	152		184	ク	216	リ	248	
25		57	9	89	Y	121	y	153		185	ケ	217	ル	249	
26		58	:	90	Z	122	z	154		186	コ	218	レ	250	
27		59	;	91	[	123	{	155		187	サ	219	ロ	251	
28		60	<	92	¥	124	¦	156		188	シ	220	ワ	252	
29		61	=	93	]	125	}	157		189	ス	221	ン	253	
30		62	>	94	^	126	~	158		190	セ	222	゛	254	
31		63	?	95	—	127		159		191	ソ	.223	゜	255	

【図211】 文字のドット構成



ます。例えば，PC-8801 では GRPH キーと，アルファベットのOを押すと，♥のマークが出ます。この♥マークは，キャラクタコード表の233番目に定義されているので，CHR＄(233) でも求められます。

PC-8201 では，グラフィックキャラクタは，◀ ↵ ▒ の3つしか定義されていませんが，キャラクタジェネレータで文字をつくって，CHR＄ 関数または GRPH キーと併用して，出力することができます。

PC-8201 では，各文字を横6ドット，縦8ドットで構成しています（図211）。

自分で新しい文字を作る場合も，6×8ドットで1文字を構成します。新しい文字は次の手順で作ります。

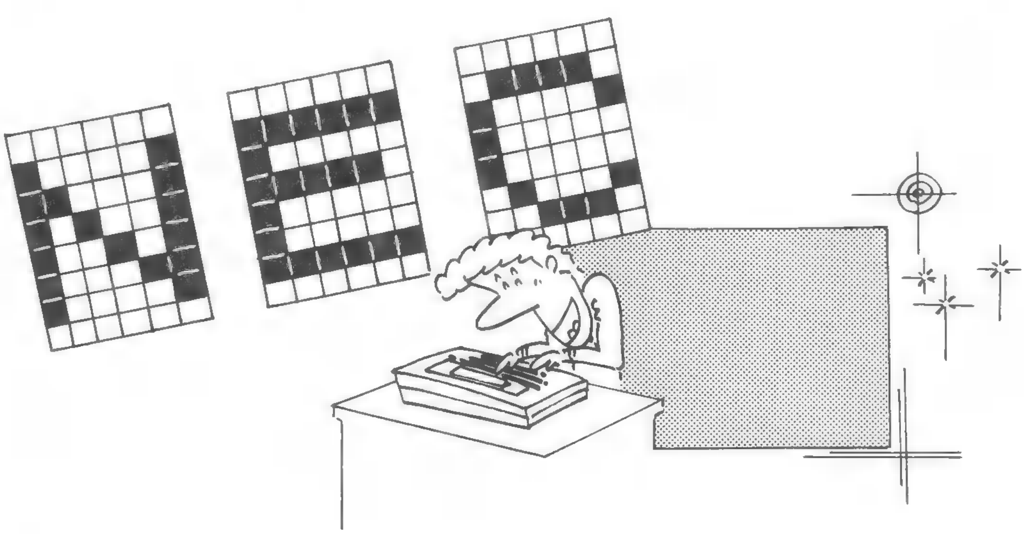
①文字のドット構成を決める。
②ドット構成を数値に変換する。
③キャラクタのデータを記録するメモリの番地を決めて，CLEAR 命令で確保する。値を POKE で書く。
④データを書き込んだ先頭番地をキャラクタ定義専用ワークエリアへ書き込んでおく。

定義された新しいキャラクタには，131から順にキャラクタコードが割り当てられます。

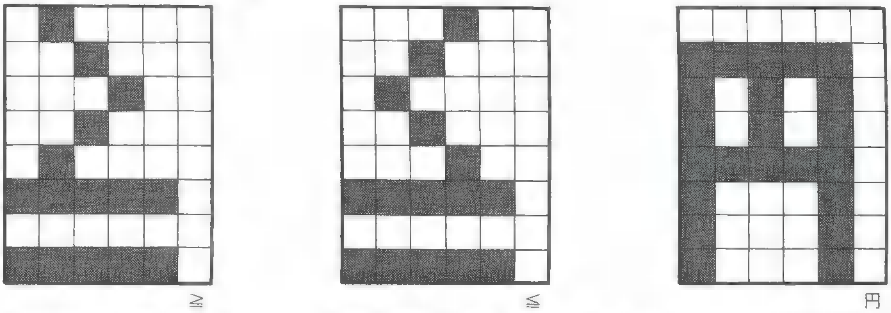
【表10】 キャラクタコード対応表

コード	対応キー	コード	対応キー	コード	対応キー	コード	対応キー
131	V	136	A	144	Q	154	@
132	B	137	S	145	W	155	¥
133	N	138	D	146	E	156	,
134	M	139	F	147	R	157	.
135	L	140	G	148	T	158	／
		141	H	149	Y	159	∧
		142	J	150	U		
		143	K	151	I		
				152	O		
				153	P		

（アルファベット下段） （アルファベット中段） （アルファベット上段） （その他）



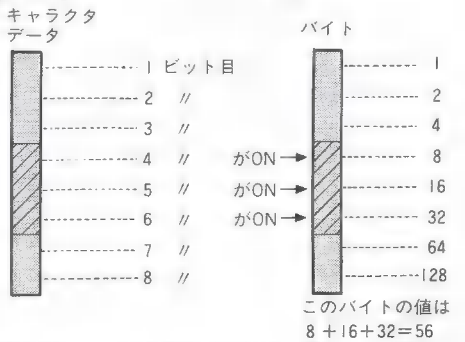
【図212】 キャラクタコードの定義の仕方



対応表のように，キャラクタコードの131～159までは，GRPHキーと何かのキーを押して出力します。224～のキャラクタコードに割り当てられた文字は，対応するキーがないので，CHR$（224）のように扱います（表10)。

では，≧，≦，円の3つの新しいキャラクタを定義します（図212)。各文字は8ドットが6列組み合わさっているので，各8ドットを1バイトのビットに対応させます（図213)。

【図213】 ドットとビットの対応



各ビットが持っている1，2，4，8，16，32，64，128の値をONのところだけ加算します。これを6回計算すると，1文字分です。これを，POKE命令で直接メモリに書き込み，さらに書き込んだ場所をワークエリアへ登録します。

≧，≦，円を定義するプログラム例をあげます（図214)。

【図214】

```
10 CLEAR 256,60000!
20 AD=60000!
30 FOR J= 1 TO 6*3
40 READ A:POKE AD,A
50 AD=AD+1
60 NEXT J
70 POKE 65216!,234:POKE 65215!,96
80 END
90 DATA 160,177,170,164,160,0
100 DATA 160,164,170,177,160,0
110 DATA 254,18,30,18,254,0
```

オートパワー機能

PC-8201では，10分以上キー入力がなされないと，自動的に電源をOFFにしてしまうオートパワーオフ機能があることをすでに説明しましたが，BASICの命令でこの機能

をコントロールすることができます。

オートパワー機能をコントロールする命令をPOWERといいます。POWERには，3つの形式があります。

①POWER タイマー
②POWER OFF または POWER OFF, RESUME
③POWER CONT

タイマーは10～255の値で指定します。タイマーの1は6秒に当たりますから，10と指定すると，60秒後に電源がOFFになります。コールドスタート時は，POWER 100（10分）に設定されています。

POWER OFF とすると無条件に電源をOFFにします。RESUME付きのPOWER OFFでは，再びスイッチを入れた時，OFFの前の状態が復元されます。

POWER CONTでは，電源OFFの機能を停止させます。電源にアダプタコードを使っている時などに使います。

POWERの指定は，一度行うと次にもう一度POWER命令で変更されるまで変化しません。BASIC以外のモードでもこのオートパワーオフ機能は有効です。

第3章

TEXT



1 TEXTの機能

PC-8201は，文章の作成編集を行うソフトウェアをTEXTの名で内蔵しています。TEXTは，キーボードから入力した文字をRAM上にTEXTファイル（.DOがつきます）として保存し，挿入・削除・訂正・置換・検索・印刷などが簡単にできるワードプロセッサのプログラムです。これらの機能を簡単に文書の編集機能と呼ぶことにします。

しかし，単に文書を編集できるプログラムが，パソコン機能から独立して存在しているのではありません。TEXTで作成したTEXTファイルは，BASICモードやTELCOMモードでデータとして処理できます。TEXTファイル内の文章が，BASICプログラムの形式になっていれば，BASICモードからロードして使うこともできます。また，逆にBASICモードで作成したプログラムおよびデータファイルをTEXTモードで処理することも可能です。

具体的な活用方法としては，まず，ワードプロセッサとして，手紙や会議資料などの文書作成ができます。文書の先頭・終端，画面の先頭・終端などにカーソルを素早く移動するための機能が豊富にあり，単語のつづりが2行に分割されないように自動的にまとめたり（ワードラップ機能といいます），必要な単語を文書の中から探したりする機能もあるので，一度作成した文書への追加修正は簡単にできます。

また，検索機能を生かして，住所録や，早見表，英単語帳などに使うことができます。検索には捜したい単語を入力するのですが，必ずしも正確である必要はなく，例えばPC-8201の単語を探す場合，PCだけでも，82だけでも一致するものを探し出します。データが特に何かの順に並んでいる必要はありません。データの量が多い時は，普通の住所録の何倍ものスピードで検索できるでしょう。

そのうえ，TEXTで作成した文書は，TEXTファイルとして自動的にRAM上に書き込まれますが，これはBASICモードでファイル処理のプログラムにより作られるものと変わりがありません。つまり，TEXTファイルはどちらのモードでも作成でき，いったんRAM上に作成されたファイルはどちらのモードでも使えるわけです。



これを利用すると，BASIC モードで作成したファイルに対し，追加・削除・修正などを行いたい時に，保守のためにプログラムを準備しなくても，TEXT モードで簡単に処理できることになります。また，一回限りのテストデータを作成するような場合も，プログラムを使わずに TEXT モードで作成することができます。

どちらのモードで作成されたファイルであっても，どちらのモードでも使えるというのは，ファイルの内容がプログラムであっても同じです。

すなわち，BASIC モードで作成したプログラムをアスキー形式でセーブすれば TEXT ファイルになりますから，TEXT により編集できますし，TEXT モードで BASIC 命令を入力し，アスキー形式のプログラムとして RAM 上に書き込み，BASIC モードでロードすることもできます。

また，BASIC モードから EDIT 命令により TEXT モードに入る場合は，まだ RAM 上へ書き込まれていないプログラム，つまり BASIC 領域のプログラムを編集することもできます。

PC-8201 の TEXT には，これらの機能のためのコマンドが用意されています。

② TEXTの使い方

TEXTの起動

TEXT モードに入るためには，①メニュー画面から TEXT を選択する，②メニュー画面から .DO 付きのファイル（TEXT ファイル）を選択する，③ BASIC モードから移るの 3 つの方法があります。

まず，メニュー画面から TEXT を選択してみましょう。

メニューモードで TEXT へカーソルを移動します（図215）。

[⏎] キーを押すと，TEXT モードに入り，画面は図216のように変わります。

「File to edit?」(編集すべきファイルは?）と，TEXT が聞いてきているので，ファイル名を入力します。ファイル名は，6 文字以内とし，後ろに拡張子として .DO をつけます。.DO は入力の時は省略してもかまいません。自動的に .DO がつけられて，RAM 上に登録されます。.DO 以外の拡張子は使えません。

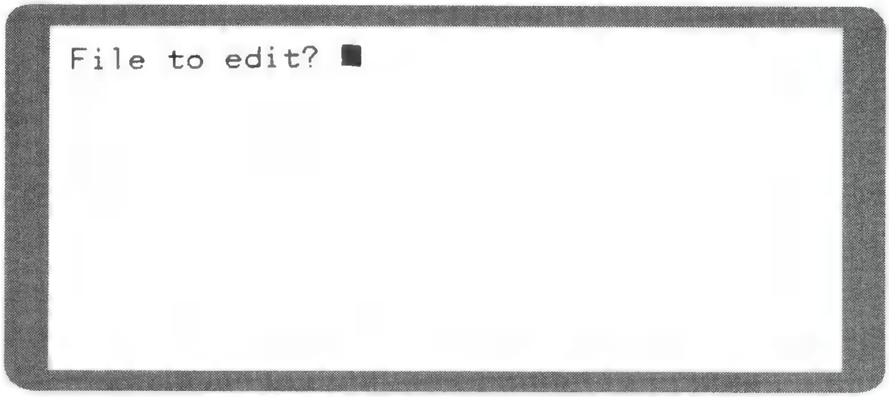
入力したファイル名が現在 RAM 上にな

【図215】

```
1983/05/04 09:48:02      (C) Microsoft #1
 BASIC      TEXT       TELCOM      -.-
 -.-        -.-        -.-         -.-
 -.-        -.-        -.-         -.-
 -.-        -.-        -.-         -.-
 -.-        -.-        -.-         -.-
 -.-        -.-        -.-         -.-
Load     Save     Name     List     28390
```

●TEXT

【図216】



ければ，直ちに文書が入力できる状態になります。RAM上にあるファイル名を入力すると，そのファイルの内容が画面に表示され，そののち入力できる状態になります。

では，ためしにsampleというファイル名を入力して［⏎］キーを押してみてください（図217）。

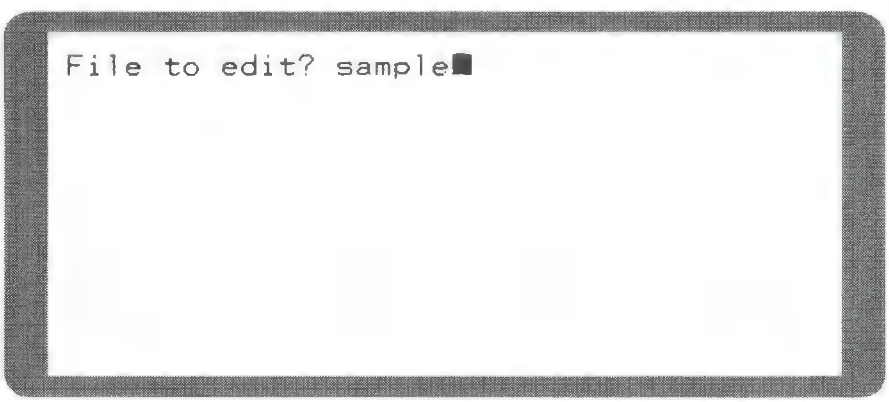
画面はクリアされ，左上にカーソルが移動します。TEXTではカーソルは■ではなくアンダーライン(—)です。また，◀のマークは文書の終わりを表しています（図218）。

文書の入力状態になっていますので，何か文字を入力してみましょう（図219）。

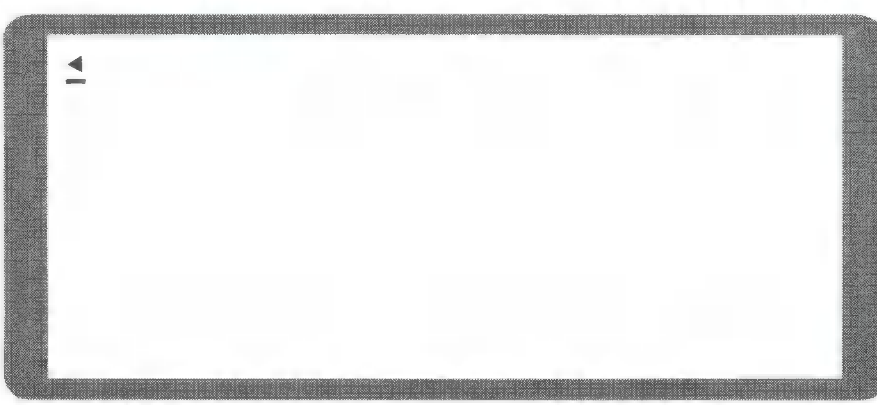
TEXTからメニューに戻るためには，ファンクションキーの10番（［SHIFT］キーと［f·5］）を押します。［ESC］キーを2回押した場合もメニューへ戻ります（図220）。

TEXTモードに入るもう一つの方法は，メニュー画面からTEXTファイルを選択する方法です。

【図217】



【図218】



【図219】

```
Personal Computer←┘
PC-8201←┘
Basic◀
```

【図220】

```
1983/05/04 09:51:10     (C) Microsoft #1
 BASIC      TEXT       TELCOM     sample.DO
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
 -.-        -.-        -.-        -.-
Load    Save    Name    List     28324
```

現在 sample の名の TEXT ファイルが作られていますので，カーソルを移動して [↵] キーを押してください。画面に SAMPLE の内容が表示されます。これは，TEXT を選択した後，FILE TO EDIT? の質問に既存のファイルの名前 SAMPLE を入力した場合と，まったく同じ結果になります（図221）。

カーソルを ◀ まで移動して続きを入力します。



【図221】

```
Personal Computer←┘
PC-8201←┘
Basic◀
```

●TEXT

TEXT モードを選択した後，ファイル名を入力した場合でも，直接，TEXT ファイルを選択した場合でも，TEXT モードに入った後の処理はまったく同じです。

では，[f·10] でメニューモードへ戻してください。

TEXT モードには，BASIC モードから入ることもできます。

まずメニューモードでBASICを選択してください。続いてプログラムを入力します(図222)。

入力が終わった後，[E][D][I][T][⏎] または [f·6] を入力します(図223)。EDIT命令については，第2章 N_{82}-BASICでも説明しましたが，BASICモードからTEXTモードへ移る命令です。このEDITでTEXTモードに入った場合は，[f·10] を押した時，または[ESC]キーを2回押した時，メニューモードではなく，元の BASIC モードへ戻ります（図224）。

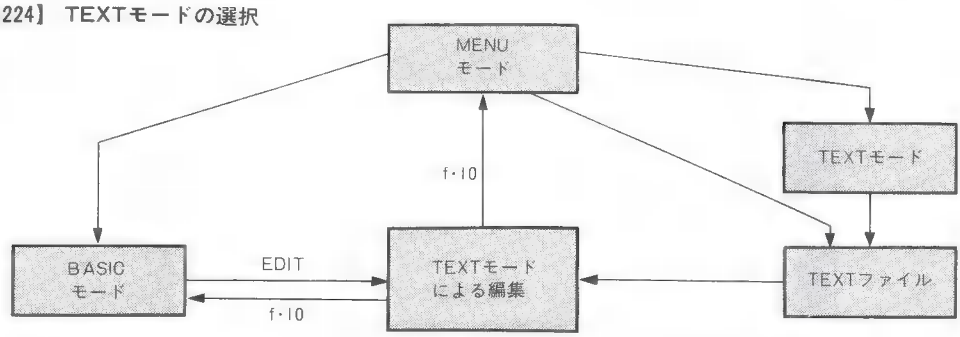
【図222】

```
NEC PC-8201 BASIC Ver 1.0 (C) Microsoft
28324 Bytes free
Ok
10 input n$
20 print n$
30 end
Edit ■
Load "  Save "  Files  List    Run
```

【図223】

```
10 INPUT N$←
20 PRINT N$←
30 END←
◀
```

【図224】 TEXTモードの選択



TEXTモードでの入力

メニューモードからTEXTモードを選択してください。FILE TO EDIT? に対し，前に作成したTEXT ファイルの名前であるsample .DOを入力してください。つづりをまちがえると，新しいファイルとみなされますので，正しくタイプします。画面にはsample .DOの内容が表示されます。

どのモードからであってもTEXTモードに入った時は，カーソルは必ず左の一番上の位置にあり，常に入力できる状態になっています（図225）。

この状態で入力をしますと，カーソルの前に文字が挿入されます（図226）。

カーソルを3行目に移動して入力してください。同じように挿入になります（図227）。

このようにTEXTモードでは，通常のま

【図225】

```
Personal Computer↵
PC-8201↵
Basic◀
```

【図226】

```
Handheld Personal Computer↵
PC-8201↵
Basic◀
```

【図227】

```
Handheld Personal Computer↵
PC-8201↵
N82Basic◀
```



まで挿入状態になっています。

TEXT モードにおいて，不要な文字の削除には，[DEL/BS] キーを使います。このキーの使い方は BASIC モードの時と同じです。直接 [DEL/BS] キーを押すと DEL（デリート）が働いてカーソルの直前の文字が消され，文章全体が前に移動します。[SHIFT] キーと [DEL/BS] キーを押すと BS（バックスペース）が働いてカーソルのある文字が消され，文章全体が前に移動します。

TEXT モードでは，常に挿入状態にあるため，入力ミスした文字を直接正しい文字に変更することはできません。

仮に，PRINT の文字を input に修正するために，カーソルを P に重ねて i を入力しても P の代わりに i になるのではなく P の前に i が挿入されます。そのため，正しい文字，ここでは input を挿入し，その後に不要な文字，ここでは PRINT を [DEL/BS] キーにより削除します（図228）。

TEXT モードで入力する時は，1 行40字を超えると自動的に改行します。タイプされた文字は，自動的にすべて RAM 上のファイルへ保存されますから，特に [⏎] キーを押す必要はありません。

[⏎] キーは，TEXT モードでは入力の区切りではなく，単に行の途中で改行することを意味します。[⏎] キーを押すと，文字のキーを押した時と同じように画面上にも ↵ マークが表れ，カーソルは次の行の最初に移ります。画面上の ↵ マークは，外の文字と同じように削除したり，挿入したり，移動したりすることができ，それにつれて，2 つの文章がつながったり，分かれたりします。

TEXT モードにおいても，画面の最終行までカーソルが進み，さらに入力しようとす

【図228】 文字の修正

```
10 PRINT "NAME=";N$
 ↓            訂正したい箇所にカーソルを移動
10 iPRINT "NAME=";N$
 ↓            前の文字は消えない
10 inputPRINT "NAME=";N$
 ↓            正しい文字を入力する
10 inputPRINT "NAME=";N$
 ↓            [SHIFT] と一緒に [DEL/BS] キーを押す
10 input_"NAME=";N$
              [SHIFT] と一緒に [DEL/BS] キーを押すとカーソルのある文字が消える
```

ると，BASIC モードにおいてと同じようにスクロール（画面の文字が1行単位で繰り上がること）が起きます。逆にカーソルが一番上の行にある時に，[↑] または [←] でカーソルをより前へ戻そうとすると，逆のスクロールが起こります。

TEXT モードには，自動ワードラップ（WORD-WRAP）機能があります。これは入力時に，1つの単語が2行に分かれないように自動的にまとめてくれる機能です。

次の単語は N_{82}-BASIC の単語ですが，空白で区切りながら順に入力してください。

PRINT　INPUT　LET　IF　THEN　ELSE　GOTO　END

カーソルは行の右側にきていますが，まだ右端ではありません。次に LPRINT を入力します。[l] [p] を入力したところでカーソルは次の行へ移ります（図229）。

続いて [r] [i] [n] [t] と入力すると，前の行にあった lp が rint の直前に移動します（図230）。

このように，空白や記号で区切られていない文字は1つの単語と考えて，2行に別れた時は自動的に1つにまとめるのがワードラップの機能です。ワードラップ機能で単語の区切りとみなしているのは，次の文字です。

（　）〈　〉〔　〕＋－＊／　空白

ただし，EDIT 命令により，BASIC モードから TEXT モードに入った場合は，ワードラップ機能は働きません。

キーボード上の [TAB] と書かれたタブキーは，BASIC モードと，TEXT モードでは働きが異なります。BASIC モードでは，[TAB] は，カーソルの水平移動を行う機能です。PC-8201 では画面上の8桁ごとにタブストップ（水平移動が止まる位置）を設定しているため，[TAB] キーを押すと，カーソルは右側で一番近いタブストップまで進みます

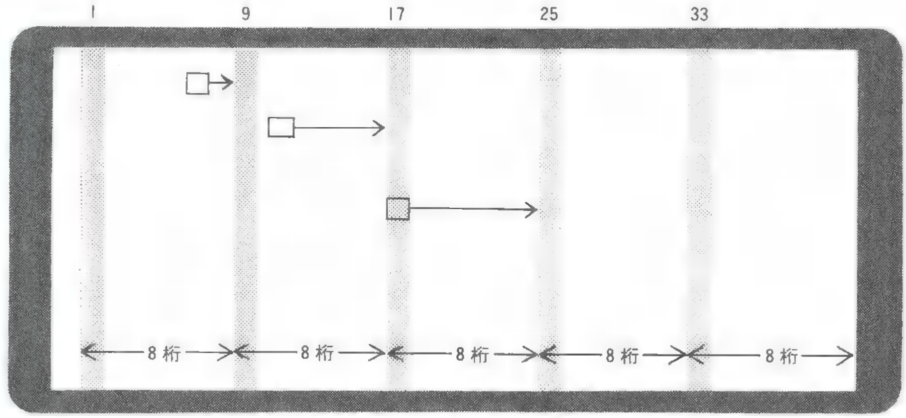
【図229】

```
print input let if then else goto end lp
◀
```

【図230】

```
print input let if then else goto end
lprint◀
```

【図231】

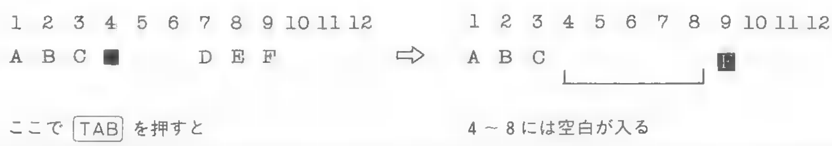


（図231)。

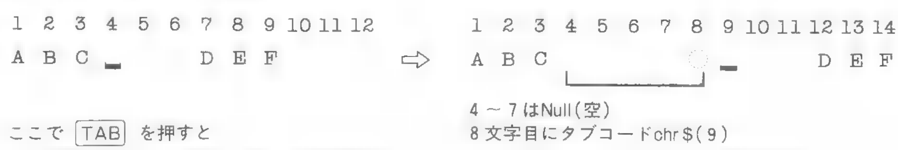
この時，最初のカーソルの位置から，タブストップまで空白（スペース）が入力されたものとみなされますので，最初のカーソル位置の右側にあった文字は消えます（図232)。

TEXT モードにおける TAB は，8桁ごとのタブストップまでカーソルを水平移動させる点では同じですが，カーソルの右側にあった文字はタブストップの後へ移動します。TEXT モードでは常に挿入状態にあるからです。また， TAB によりタブセットまでカーソルは進みますが，空白が入力されるので

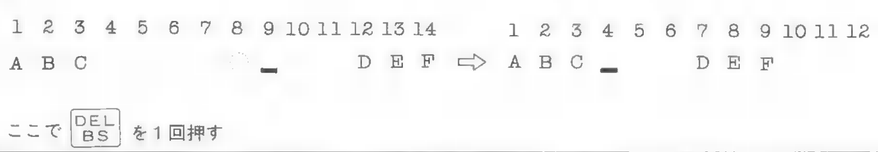
【図232】 BASICモードのTABキー



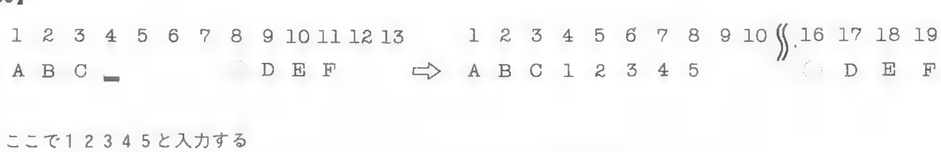
【図233】 TEXTモードのTABキー



【図234】



【図235】



はなくヌル（空）の状態です（図233）。

TAB キーが押されて，タブセットの位置にカーソルが進んだ時は，画面上には何も表れていませんが，タブコード（アスキーコードの9）が入力されています。

DEL/BS キーを1回押して，8桁目のみえないタブコードを削除してみます。タブコードを削除するとタブセットより後の文字はタブが押される前の状態に戻ります（図234）。

また，タブセットを超えて文字を挿入すると，全体に次のタブセットまで移動します（図235）。

TAB を使うと，文字の桁ぞろえが簡単にできます。

③ TEXTのコマンド

TEXT モードには，その機能を生かすための特有の命令がいくつかあります。これらは TEXT コマンドといいます。

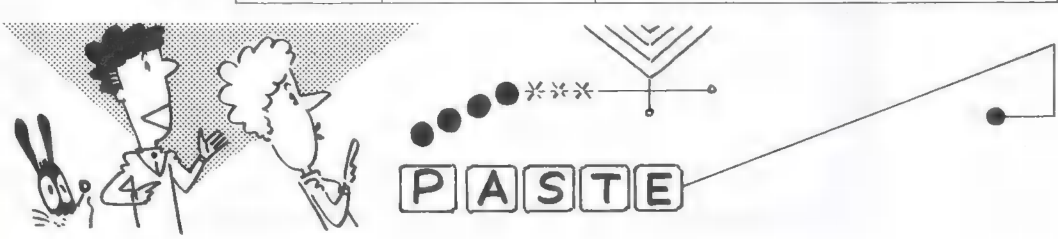
TEXT モードでは，文字の入力はすべて自動的に TEXT ファイルとして処理されるため，一般キーからコマンドを入力することができません。そのため，TEXT のコマンドはすべて，ファンクションキーと，PAST/INS キーに登録されています（表11）。

TEXT モードでは，PASTE（ペースト）バッファと呼ばれる特別な作業領域が使えます。

PASTE バッファは，メニュー画面には表れませんが，RAM 上に作成される領域で，テキストファイルの一部分を取り出すために使う特別な機能を持ちます。

【表11】 TEXTコマンド一覧表

コマンド	キー	機能
FIND	f·1	ストリング（文字列）の検索
NEXT	f·2	次のストリングの検索
SELECT	f·3	PASTEバッファへ転送する領域の指定
CUT	f·4	指定の領域をPASTEバッファへ転送，画面から消す
COPY	f·5	指定の領域をPASTEバッファへ転送，画面から消さない
PASTE	SHIFT + PAST/INS	PASTEバッファの内容をファイルへ転送
KEYS	f·6	コマンド名の表示ON／OFF
MENU	f·10	MENUへ戻る



FIND

TEXT ファイル中から特定の単語（ストリング）を探すのが FIND コマンドの機能です。FIND は f·1 に登録されています。

いま，TEXT ファイルの内容が図236のようになっているとします。

ここで，f·1 を押してFINDコマンドを実行します。画面一番下の行に「String:」の文字が表れます（図237）。

ここで探したい単語（ストリング）を入力します。いま100点の人を探したいので，100と入力します。ストリングは24文字まで指定でき，それ以上は入力しても，ピッという音がして先へ進みません（図238）。

ストリングを入力して ⏎ キーを押すと，現在カーソルのある単語の次から一致する単語を探します。見つかると，カーソルをその単語の先頭へ移動します（図239）。

みつからないとカーソルの位置はそのままで，「No match」の文字が表示されます。

指定したストリングは，もう一度ストリングを指定しなおすか，TEXT モードを抜けるまで変わりません。

【図236】

```
1,ｱﾍﾞ,95
2,ｲﾄｳ,62
3,ｲﾏｲ,77
4,ｳｴﾀﾞ,100
5,ｴﾝﾄﾞｳ,86
6,ｵｶﾀﾞ,82
7,ｵﾉ,100
8,ｶﾜﾑﾗ,100
9,ｷﾀ,60
10,ｸﾎﾞ,73
11,ｸﾜﾀ,81
12,ｺﾊﾞﾔｼ,68
13,ｻｶｲ,66
14,ｻｶﾓﾄ,77
15,ｼﾏﾀﾞ,69
```

【図237】

```
1,ｱﾍﾞ,95↵
2,ｲﾄｳ,62↵
3,ｲﾏｲ,77↵
4,ｳｴﾀﾞ,100↵
5,ｴﾝﾄﾞｳ,86↵
6,ｵｶﾀﾞ,82↵
7,ｵﾉ,100↵
String:■
```

【図238】

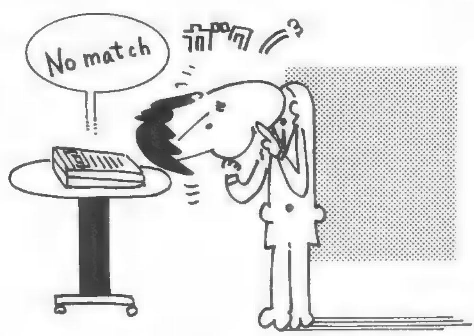
```
1,アベ",95↵
2,イトウ,62↵
3,イマイ,77↵
4,ウエダ",100↵
5,エンド"ウ,86↵
6,オカダ",82↵
7,オノ,100↵
String:100↵
```

【図239】

```
1,アベ",95↵
2,イトウ,62↵
3,イマイ,77↵
4,ウエダ",100↵
5,エンド"ウ,86↵
6,オカダ",82↵
7,オノ,100↵
8,カワムラ,100↵
```

NEXT

NEXT コマンドは，FIND で指定した単語を，現在カーソルのある単語の次の単語から探す機能で，f·2 に登録されています。みつかるとカーソルをその単語の先頭へ移動し，みつからないとカーソルの位置はそのままで「No match」の文字が表示されます。



NEXT は，通常 FIND に続けて使います（図240）。

【図240】

```
1,アベ",95↵
2,イトウ,62↵
3,イマイ,77↵
4,ウエダ",100↵
5,エンド"ウ,86↵
6,オカダ",82↵
7,オノ,100↵
8,カワムラ,100↵
```

SELECT

SELECT コマンドは，TEXT ファイルのどの部分を PASTE バッファへ転送するかを指定します。SELECT は f·3 に登録されています。

SELECT コマンド，すなわち f·3 を使う時は，TEXT ファイルを画面に出し，抜き出したい部分の最初までカーソルを移動します。

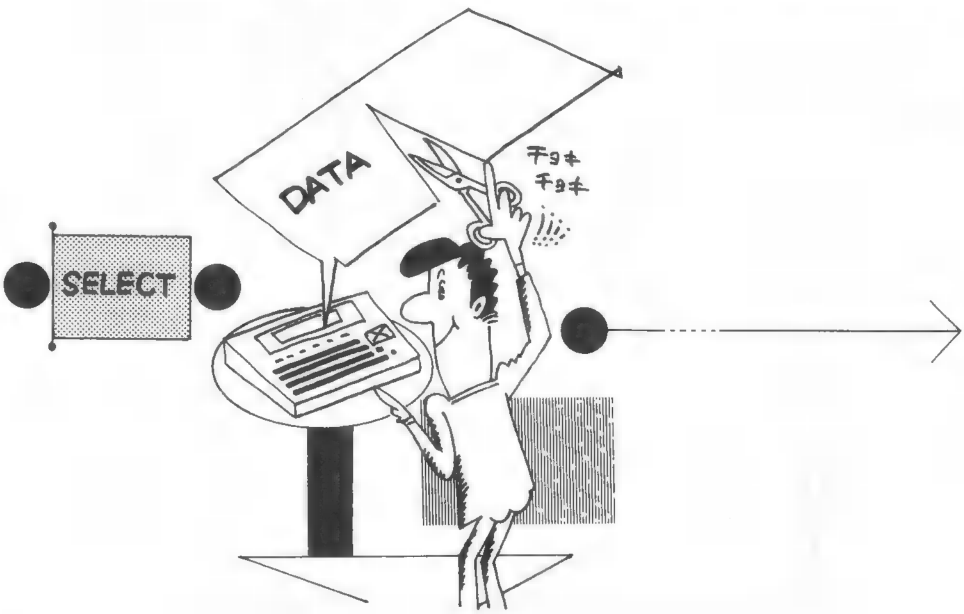
f·3 を押すと，そこがPASTE バッファへ転送するために TEXT ファイルから抜き出す領域のはじまりになります。その後，カーソルを移動すると，移動した部分がすべてリバース（白黒反転）状態になります（図241)。このリバース状態の部分が，SELECT された領域を意味します。

この領域の指定は，メニューモードに戻ったり，CUT や COPY のコマンドを実行すると解除されます。

また，仮に途中で中止したい場合には，STOP キーを押すと，リバース状態が元へ戻り，SELECT コマンドがキャンセルになります。そのほか ⏎ キーや TAB キーでも SELECT コマンドはキャンセルすることができます。

【図241】

```
1,ｱﾍﾞ,95↵
2,ｲﾄｳ,62↵
3,ｲﾏｲ,77↵
4,ｳｴﾀﾞ,100↵
5,ｴﾝﾄﾞｳ,86↵
6,ｵｶﾀﾞ,82↵
7,ｵﾉ,100↵
8,ｶﾜﾑﾗ,100↵
```



CUT

CUT コマンドは SELECT で指定された TEXT ファイルの一部分をファイル上から削除し，PASTE バッファへ転送する機能を持ちます。

CUT コマンドは [f·4] に登録されています。

CUT コマンド，すなわち [f·4] を押すと，現在リバース状態になっている部分がファイルおよび画面から消え，PASTE バッファにこの削除分が転送されます。

COPY

COPY コマンドは，SELECT で指定された TEXT ファイルの一部分を，ファイル内容は，そのままにして，PASTE バッファへ転送します。COPY は [f·5] に登録されています。

TEXT ファイルの一部が SELECT されている状態，すなわち一部がリバースされている状態で [f·5] を押します。PASTE バッファへ転送が終わると，リバースしていた部分は正常へ戻ります。

PASTE

PASTE は PASTE バッファの内容を転送する働きを持っています。PASTE は，[SHIFT] キーを押しながら，[PAST INS] キーを押します。PASTE バッファの内容は何度使用しても消えません。

では，カーソルを図242の位置へ移動して

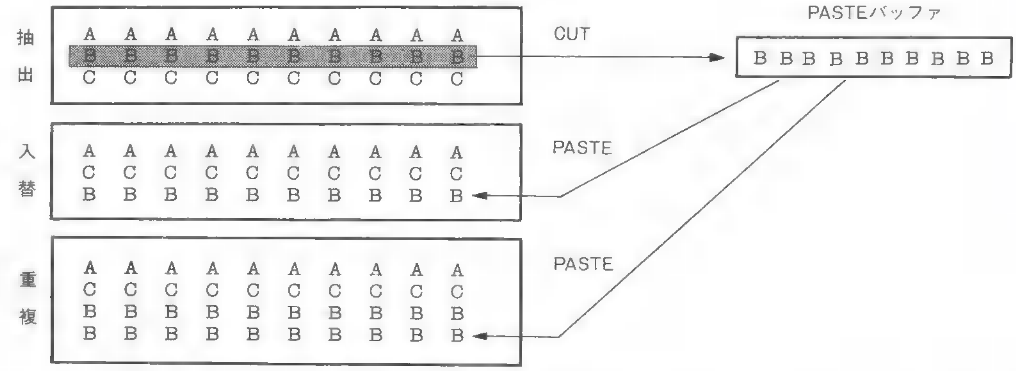
【図242】

```
7,オノ,100↵
8,カワムラ,100↵
9,キタ,60↵
10,クホ゛,73↵
11,クワタ,81↵
12,コハ゛ヤシ,68↵
13,サカイ,66↵
14,サカモト,77↵
```

【図243】

```
7,オノ,100↵
8,カワムラ,100↵
9,キタ,60↵
3,イマイ,77↵
4,ウエタ゛,100↵
5,エント゛ウ,86↵
6,オカタ゛,82↵
10,クホ゛,73↵
```

【図244】 TEXTファイルの抽出と挿入



からPASTEコマンドを実行します。

先ほどCOPYコマンドでPASTEバッファへ転送されている部分が，カーソルのある位置へ挿入されます（図243）。

このようにPASTEコマンドを使うと，TEXTファイル上のある部分とある部分を入れ替えたり，同じ内容を重複させたりすることが簡単にできます。

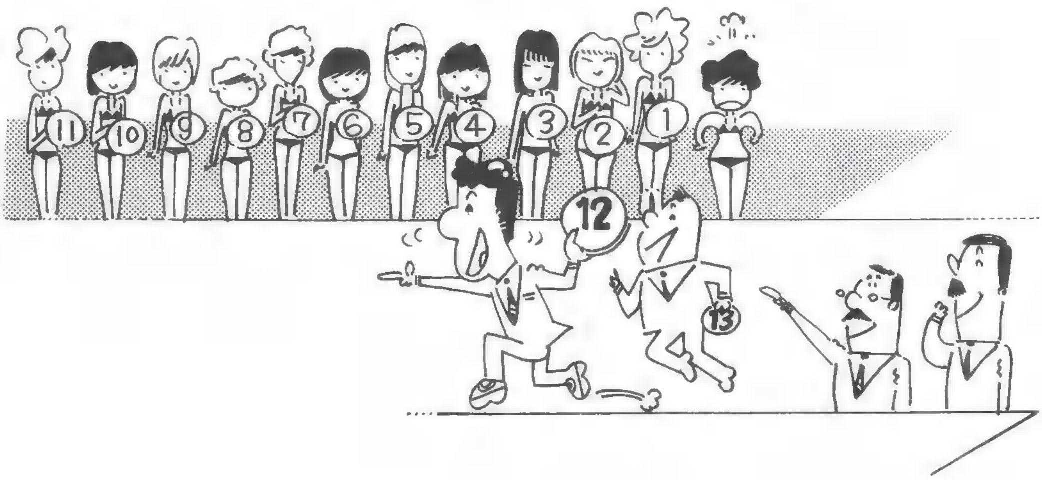
PASTEバッファは，個々のTEXTファイルとは独立した領域のため，AというTEXTファイルの一部を抽出してPASTEバッファへ移した後，BというTEXTファイルへ挿入することもできます（図244）。

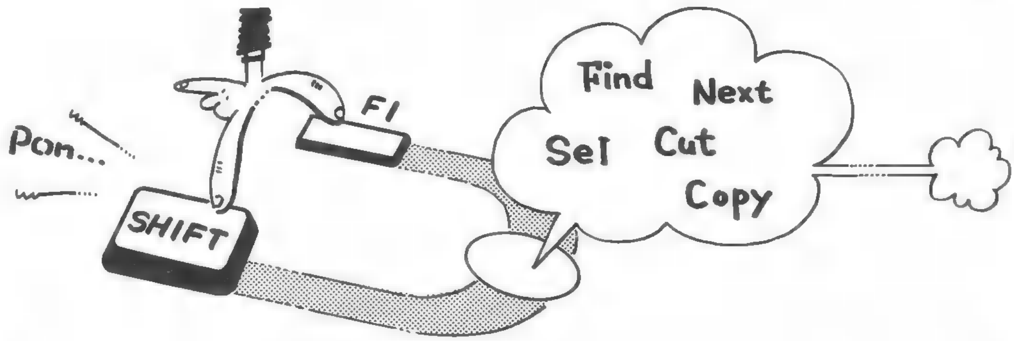
PASTEバッファの大きさには，特に制限がありません。何が記憶されているかにより，流動的に変化します。

PASTEバッファの内容は，新たな内容が入れられるまで，すなわち，CUTやCOPYが実行されるまで，変化しません。他のモードへ移ったり，電源を切ったりしても，そのまま残ります。

なお，PASTEバッファの内容を完全に消してしまいたい場合は，次の3つの方法があります。

① BASICモードでCLEAR命令を使う。
② SELECTとCUTを続けて行う（［f·3］を押した後，領域指定をせずに［f·5］をすぐ押す）。
③ SELECTとCOPYを続けて行う（［f·4］と［f·5］を続けて押す）。





KEYS

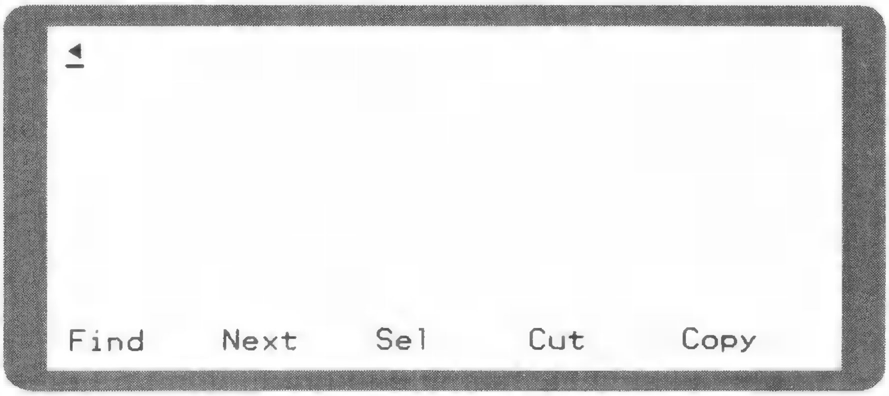
KEYS は，画面の一番下の行にファンクションキーの登録内容を表示するかどうかのコマンドです。

KEYS 自身は，ファンクションキーの6番目に登録されています。

ファンクションキーの表示は，通常はなされていません。KEYS コマンド，すなわち [f·6] を1回押して，ファンクションキーの内容，すなわち，TEXT コマンドを表示してください（図245）。

ファンクションキーの内容表示は，もう1回 [f·6] を押すと消えます。

【図245】



MENU

ファンクションキーの10番目には，他のモードの場合と同様にメニューモードに戻るための MENU コマンドが登録されています。TEXT モードの場合は，[ESC] キーを2回押した場合も，メニューモードへ戻ります。ただし，BASIC モードから EDIT コマンドにより，TEXT モードに入った場合は，[f·10] または [ESC] キーを2回押した時はメニューではなく BASIC モードへ戻ります。

④ カーソルの移動

カーソルはカーソル移動の4つのキーで上下左右へ1つずつ動きますが，TEXT モードではコントロールキーを使って，もっと効果的にカーソル移動ができます（表12）。

【表12】 TEXTのコントロールコード表

操　作	機　　能	同等の操作
CTRL + A	カーソルを左隣の単語へ移動する	SHIFT + ←
CTRL + B	カーソルを一画面分下へ移動する	SHIFT + ↓
CTRL + C	コマンド（SELECT, FIND）をキャンセルする	STOP
CTRL + D	カーソルを1文字右へ移動する	→
CTRL + E	カーソルを1行上へ移動する	↑
CTRL + F	カーソルを右隣の単語へ移動する	SHIFT + →
CTRL + H	カーソルの直前の文字を削除する	DEL BS
CTRL + I	タブ移動する	TAB
CTRL + K	NEXTコマンドを実行する	f･2
CTRL + L	SELECTコマンドを実行する	f･3
CTRL + M	⏎キーと同じ働きをする	⏎
CTRL + N	FINDコマンドを実行する	f･1
CTRL + O	COPYコマンドを実行する	f･5
CTRL + P	KEYSコマンドを実行する	f･6
CTRL + Q	カーソルを行の左端へ移動する	CTRL + ←
CTRL + R	カーソルを行の右端へ移動する	CTRL + →
CTRL + S	カーソルを1文字左へ移動する	←
CTRL + T	カーソルを一画面分上へ移動する	SHIFT + →
CTRL + U	CUTコマンドを実行する	f･4
CTRL + W	カーソルをファイルの先頭へ移動する	CTRL + ↑
CTRL + X	カーソルを1行下へ移動する	↓
CTRL + Z	カーソルをファイルの終端へ移動する	CTRL + ↓
CTRL + →	カーソルを行の右端へ移動する	CTRL + R
CTRL + ←	カーソルを行の左端へ移動する	CTRL + Q
CTRL + ↑	カーソルをファイルの先頭へ移動する	CTRL + W
CTRL + ↓	カーソルをファイルの終端へ移動する	CTRL + Z

第4章

TELCOM



1 TELCOMの機能

PC-8201は，RS-232Cインターフェイスを使用して通信を行うソフトウェアをTELCOMの名で内蔵しています。

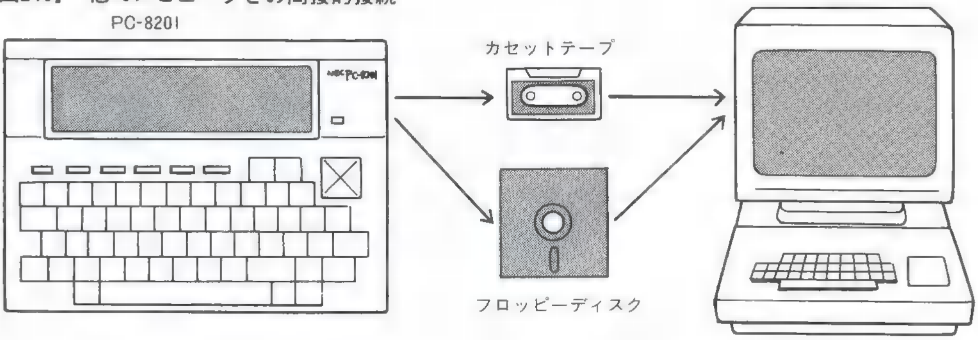
TELCOMを使うと，RAM上のファイルを他のコンピュータへ送ったり，その逆などの処理が，ファンクションキーに登録されているコマンドだけで可能です。

PC-8201は，ハンドヘルド・コンピュータですから，一定の場所に設置して使うよりも，あちこちに持ち歩いて使うことが多いはずです。周辺機器を全部一緒に携帯することは大変ですから，多くの場合，PC-8201で簡単な計算とデータの収集（RAM上に記録）だけを行い，自宅や会社へ帰ってから，データを加工したり，印刷したりすることになるでしょう。その時に，PC-8201よりも機能の高いコンピュータで処理させたいことも出てきます。この時に，いったんPC-8201から，カセットテープや，フロッピーディスクへデータを記録し，その後，この記録した媒体を別のコンピュータにセットして使うことができます（図246）。

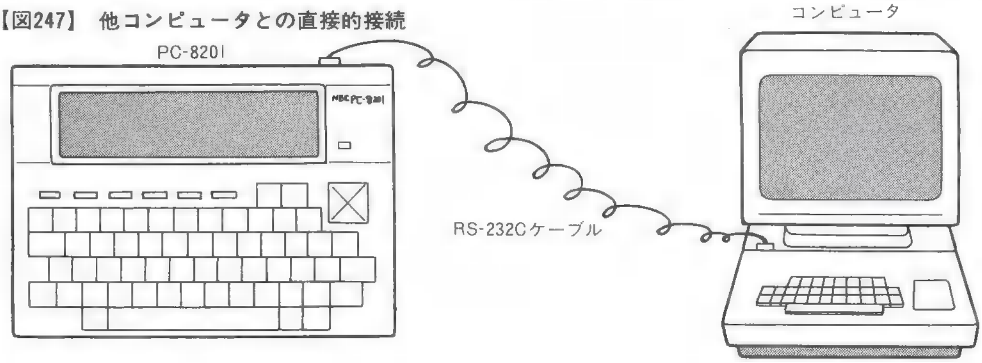
しかし，使用しているコンピュータがRS-232Cインターフェイスによる接続が行えるコンピュータであれば，直接PC-8201とデータのやり取りができます（図247）。

また，RS-232Cから取り出した信号を電話回線を使って送ると，PC-8201と遠隔地

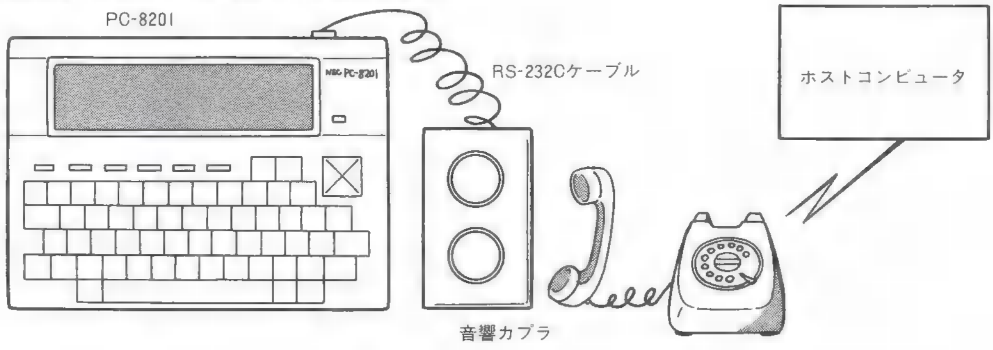
【図246】 他コンピュータとの間接的接続



【図247】 他コンピュータとの直接的接続



【図248】 コンピュータ間のデータのやり取り



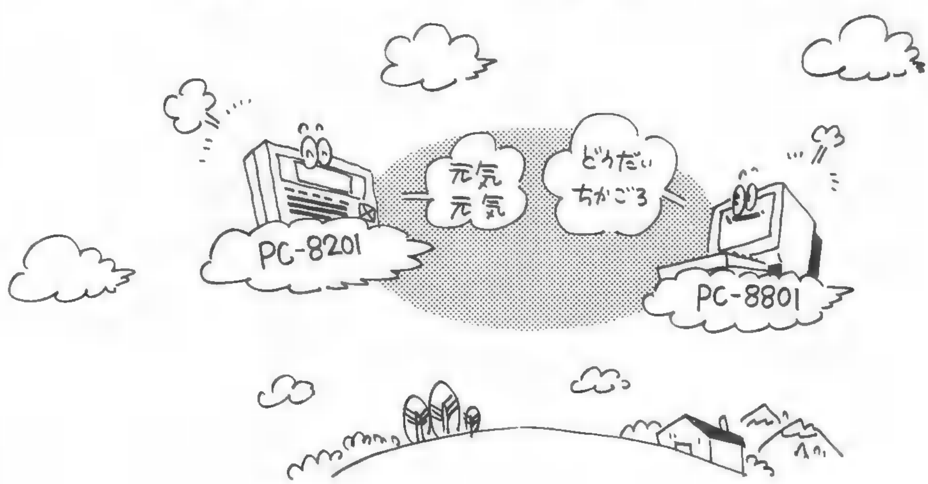
にある別のコンピュータとの間で，データのやり取りが可能になります（図248)。

ここで，RS-232C について，説明しておきましょう。

RS-232C とは，コンピュータと他の装置をつないで通信を行うための規格です。コンピュータとパソコン，パソコンとパソコンといった接続のほかに，パソコンとプリンタ，パソコンとプロッタ，パソコンとデジタイザなどの接続に用いることがあります。RS-232C の規格に沿ったインターフェイスであれば，その内部的機能はもちろん，ケーブルを接続するコネクタ部分も完全に統一されていますから，別々の装置の接続が簡単にできるのです。2つの装置が接続されると信号がシリアルに（1ビットずつ）送られます。RS-232C インターフェイスを装備していないパソコンでも，カードやボードを追加することで，機能を拡張できるようにしてあるのが普通です。

PC-8201 は，このRS-232C インターフェイスを標準装備，つまり内蔵しています。また，ソフトウェア面では，N_{82}-BASIC 内に，RS-232C による接続を前提にした命令を含んでいますから，特別な言語を覚えなくても BASIC モードのプログラムで，処理したデータを送信したり，受信したデータを加工したりできます。

さらに，TELCOM のソフトウェアでは，通信形式の指定やデータの送受信をコマンド入力だけで実行します。つまり，ON-LINE（オンライン）を行うのにプログラミ



ングが不要です。

TELCOM ソフトウェアの具体的な機能としては次のものがあります。

①メニュー画面から簡単に選択できる。

②強力な9つのコマンドを持ち，それらはすべて，ファンクションキーに登録されている（図249）。

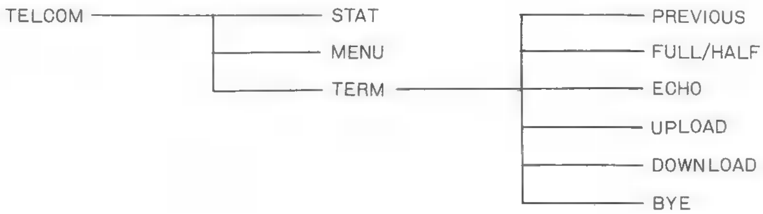
③エスケープシーケンス，コントロールコードの使用ができる。

④通信形式の設定が簡単に変更できる。

⑤キーボードからのデータ送信のほか，RAM 上のファイルの送受信ができる。

⑥送受信のデータを画面やプリンタに出力できる。

【図249】 TELCOMソフトウェアの機能



② TELCOMコマンドの使い方

TELCOM を使うために，まず，メニュー画面から TELCOM を選択します（図250）。

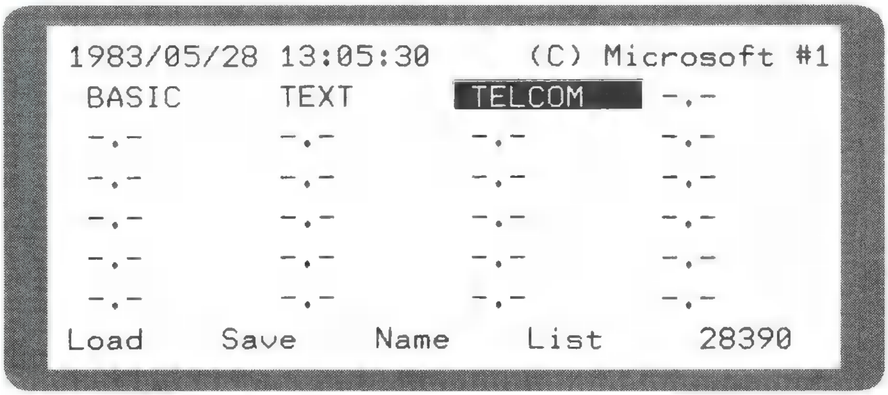
⏎ キーが押されると，画面がクリアされ，現在設定されている通信形式を表して，コマンドの入力待ちとなります。通信形式は簡単に変更ができますが，電源を切っても変化はしません（図251）。

画面に「Telcom:」が表示されている時は STAT，TERM，MENU の3つのコマンドのいずれかが入力できます。この3つのコマンドはキーボードから入力してもよいのですが，ファンクションキーに登録されています。

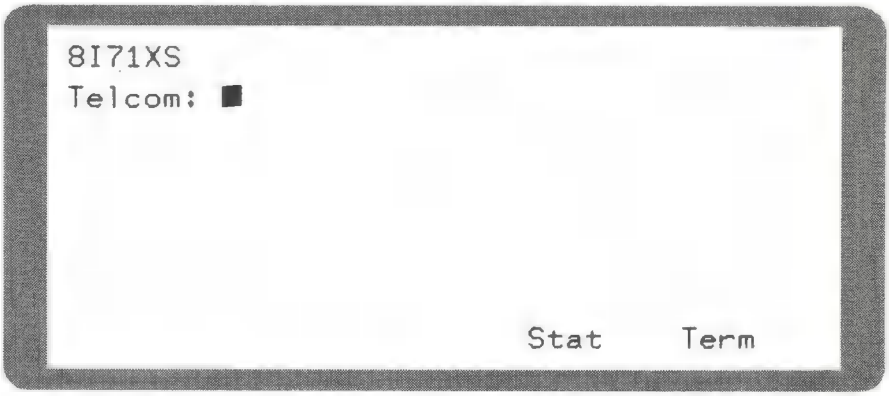
STAT f·4
TERM f·5
MENU f·10

このコマンド以外を入力した場合は，ブザー音で警告し，コマンド入力のやり直しになります。

【図250】



【図251】



STAT

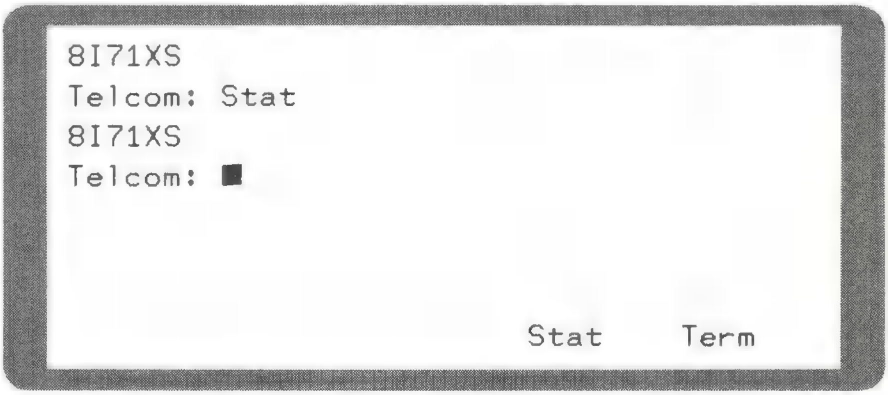
TELCOM モードの時，ファンクションキーの 4 番に登録されているコマンドで，通信形式の状態（STATUS）の表示，または変更を行います。

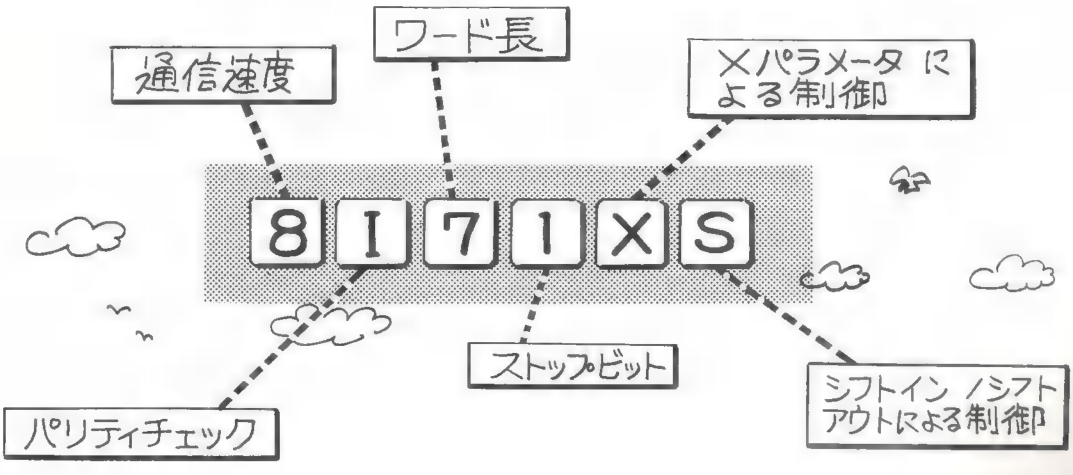
STAT [⏎] の場合，現在設定されている通信形式を表示し，コマンド入力へ戻ります（図252）。

通信形式の各桁は次の意味を持っています。

1 桁目　通信速度
2 桁目　パリティチェック
3 桁目　ワード長
4 桁目　ストップビット
5 桁目　X パラメータによる制御
6 桁目　シフトイン／シフトアウトによる制御

【図252】





【図253】

```
8I71XS
Telcom: Stat
8I71XS
Telcom: Stat 5N81XS
Telcom: ■

                       Stat     Term
```

8I71XS は，コールドスタートの時の初期値で，これ以降 STAT コマンドまたは N_{82}-BASIC の OPEN 命令で通信形式を変更すると，その値が，次に通信形式を設定するまで変わらずに残ります。

通信形式の変更を STAT で行う場合は，STAT に続けて，6桁の通信形式を入力します（図253）。

各パラメータの指定は，次のように行います。

①通信速度……接続するコンピュータや装置により選択します。

1：　75 bps
2：　110 bps
3：　300 bps
4：　600 bps
5：　1200 bps
6：　2400 bps
7：　4800 bps
8：　9600 bps
9：　19200 bps

bps＝bit per second

②パリティ……通信時に，データがまちがった信号で送られていないか，元のデータと同じもの（parity）かどうかをチェックするために，チェック用のビットを指定するパラメータです。

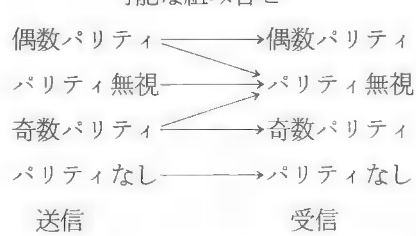
N：　パリティなし
E：　偶数パリティ
O：　奇数パリティ
I：　パリティビット無視

パリティなしとパリティ無視を混同しないでください。パリティなしは，パリティビットが送信されないことを意味し，無視は，パリティビットが何であっても受けつけることを意味します。したがって，パリティなしの設定では，パリティ付きのデータは受信できませんし，パリティ無視の設定では，パリティなしのデータは受けつけられません。パリティ無視で送信する場合は，パリティビッドを常にゼロで送るので，受信側でもパリティ無視を指定する必要があります。

ワード長が8ビットの時，パリティビット無視の指定はできません。

これをまとめると次のようになります。

可能な組み合せ



③ワード長……1文字が何ビットで構成されているかの指定です。

6：6ビット長
7：7ビット長

8：8ビット長

データ長が7ビット長の時は，シフトイン／シフトアウトシーケンスによる通信制御を行わないと，カナ文字を扱うことができません。

④ストップビット……データの区切りに使うストップビットの長さを表します。

1：1ストップビット

2：2ストップビット

⑤ X パラメータによる制御……送信のスピードに受信側が間に合わないことを避けるため，受信バッファが一杯になったら，相手側に送信を一時停止するように要求し，空になったら送信再開を要求する制御です。

X：制御を行う

N：制御を行わない

具体的には，受信時にバッファ（256文字）の残りが23文字分になると，相手側にCHR$(19)のコード CTRL ― S を送って通信の中断を要求し，空になるとCHR$(17)のコード CTRL ― Q を送って，送信再開を要求します。また，送信時に相手側から CTRL ― S が送信されてきた場合は，送信を停止し， CTRL ― Q が送られてきた場合に再開します。Xパラメータによる制御を行わない方が，送受信のスピードは速いのですが，画面表示や印刷を行いながらのデータ通信の場合，受信のスピードに，受け取ってからの処理のスピードが追いつかないことが起こります。この時，受信バッファがあふれてしまい，データの一部が欠除してしまいます。

⑥シフトイン／シフトアウトシーケンス……シフトイン（SI）コードまたはシフトアウト（SO）コードに続いてコードを出力するかどうかの指定です。

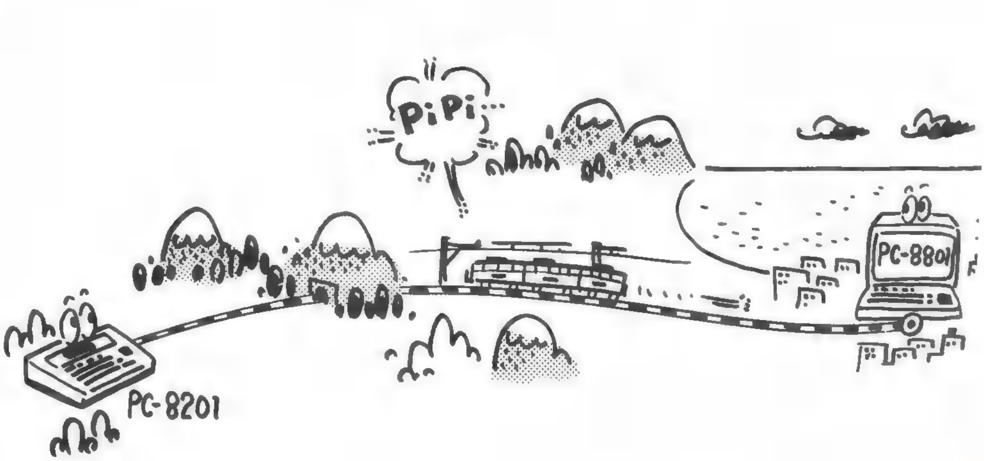
S：制御を行う

N：制御を行わない

いうまでもないことですが，どのような通信形式であっても，送信側と受信側で同じ形式の指定を行わなければなりません。例えば，送信側と受信側の通信速度が違っていれば，正しく信号を受け取ることができませんし，片方が偶数パリティを，もう片方が奇数パリティを指定していると，すべてのデータがパリティエラーになってしまいます。

MENU

TELCOM モードの時，他の BASIC モードや TEXT モードの時と同様に，ファンクションキーの10番に登録されているコマンドで，TELCOM モードからメニューモ



ードへ戻る働きをします。[f·10] には，[⏎] キーの機能が含まれていますから，[⏎] キーを押す必要はありません。

TERM

TELCOM モードの時，ファンクションキーの5番に登録されている機能で，TERMINAL（ターミナル）モードを開始する機能を持ちます。

TELCOM を選択しただけでは，実際には通信ははじまりません。TERM コマンドを実行した後，PC-8201 は RS-232C 回線とつながった状態になり，通信が可能になります（図254）。

TERM コマンド実行後は，送信するためのデータをキーボードから入力することと，ファンクションキーに登録されているサブコマンドの実行のいずれかが行われます。

①キー入力……ファンクションキー以外のすべてのキー入力が送信される。

②サブコマンド入力……ファンクションキーを使って入力する。キーボードからは入力できない。

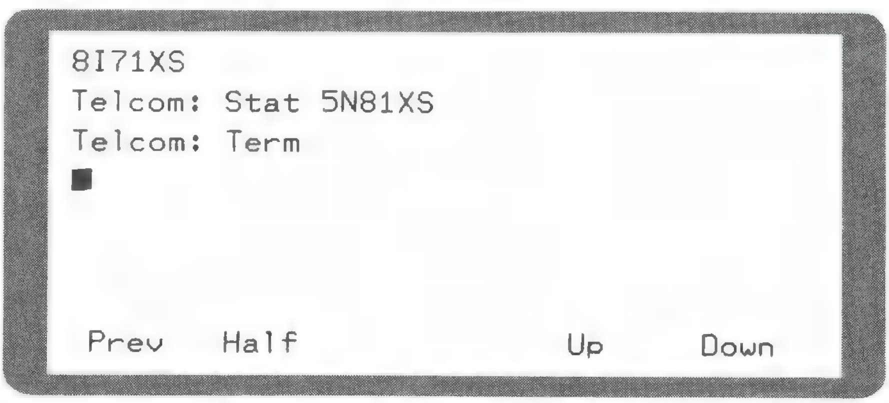
キーボードからの入力は，すべてそのまま送信されてしまうので，カーソル移動して画面上を修正しても，カーソル移動そのものもデータとして扱われてしまいます。文字の訂正には，[DEL BS] キーを使って，訂正コードそのものを送る必要があります（図255）。

PREVIOUS

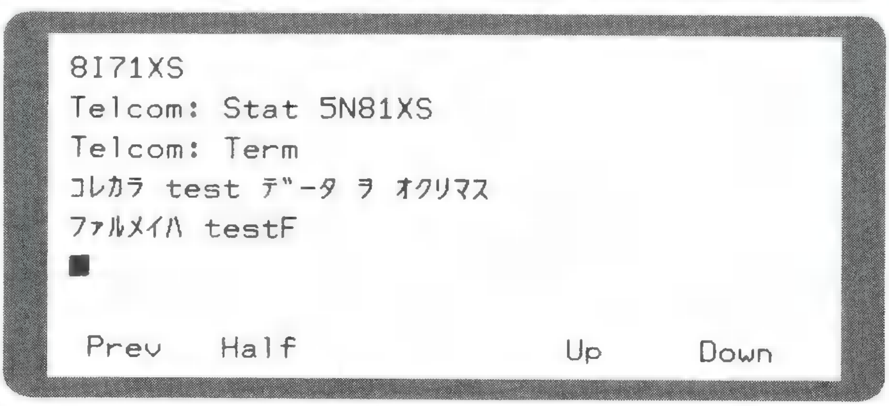
TERM モードの時，ファンクションキーの1番に登録されています。画面上では，「Prev」とだけ表示されています。スクロールによってはみだした前8行に戻って表示する機能です。

PC-8201 の TERM モードでは，画面表示のために，16行分（8行×2画面＝2ページ）の領域をバッファとは別に確保しています。画面表示の際，8行を超えるとスクロ

【図254】



【図255】



ール（1番下に新しい行を入れるために1番上の行から消していく）が起きますが，この時，直前の8行分つまり前のページが領域に残っているため，［f·1］を押していつでも画面を戻すことができます。1ページ目を表示している間も，データの受信は中断せずに行われています。ただしバッファにためるだけで1ページ目を表示しているので，当然のことながら画面には表示しません。

もう一度［f·1］を押すか，キーボードからの入力を再開すると，2ページ目の画面に戻ります。

HALF／FULL

TERM モードの時，ファンクションキーの2番に登録されているコマンドで，半二重／全二重の切り替えを行います。［f·2］を押すごとに，HALF と FULL が交互に切り替わります。

全二重，つまり FULL が選択されている時は，PC-8201 が送信したデータを相手側が折り返し転送してくるので（エコーバック），これを画面に表示します。半二重，つまり HALF が選択されている時は，PC-8201 は，送信するデータを自分自身で画面表示（セルフエコー）し，相手側から返送されなくても，何を送信しているのかわかるようにしています。

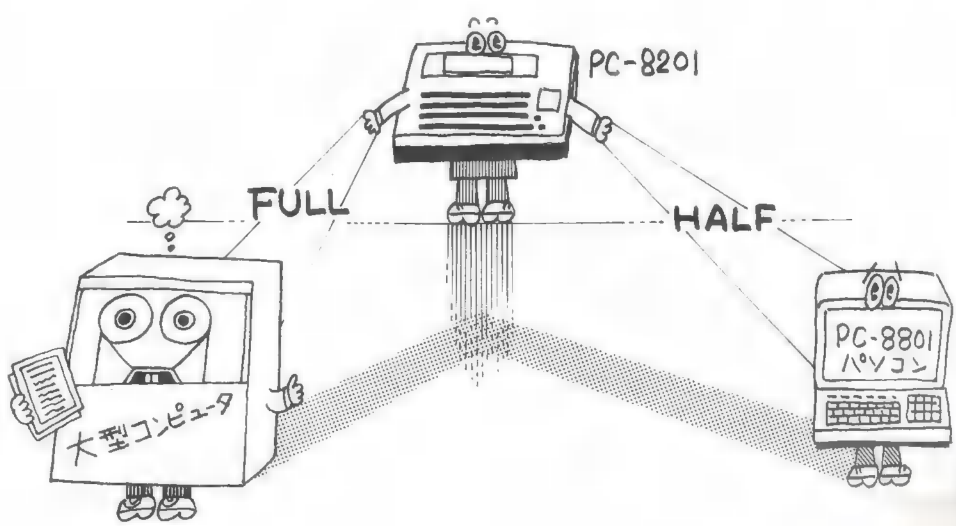
大型コンピュータのターミナルとして PC-8201 を使用する場合は FULL を，他のパソコンと接続する時は HALF を選択してください。

ECHO

TERM モードの時，ファンクションキーの3番に登録されているコマンドで，受信したデータをプリンタへ送る働きをします。

TERM モードに入った時点では，ECHO 機能は選択されず，ファンクションキーの3番目の表示も空白になっています。［f·3］を押すと，ECHO の文字が表示され，もう一度，［f·3］を押して ECHO の機能を中止するまでプリンタへデータを送ります。

通信速度の設定にもよりますが，通常，通信速度よりもプリンタの印字速度の方が遅いため，受信しながらデータを印刷すると，送られてくるデータの方が滞りすぎることになります。256バイトの受信バッファを超えてしまうと，超えたデータが，消失してしまうことにもなります。Xパラメータで，バッファが一杯になりかけたら，送信側へ，送信停止の要求を出すように設定するか，受信しな



からの印刷をやめて，とりあえずRAM上のファイルへ記録し，後で印刷する方法をとった方がよいでしょう。

UPLOAD

TERMモードでファンクションキーの4番に「Up」と登録されているコマンドで，RAM上のファイルを送信する働きをします。

[f･4]を押すと，PC-8201は，実行中のTERMモードを一時中断し，RAM上のファイルを送信します。まず，送信したいファイル名を指定します（図256）。

送信できるファイルはRAM上のTEXTファイル（.DO付き）だけです。BASICプログラムを送信したい場合は，アスキーセーブを行って，TEXTファイルの形にしておきます。拡張子.DOは省略できます。

ファイル名が正しく入力されると，[⏎]キーを押した直後から送信が始まります。送信中は，「Up」の文字が反転しています（図257）。

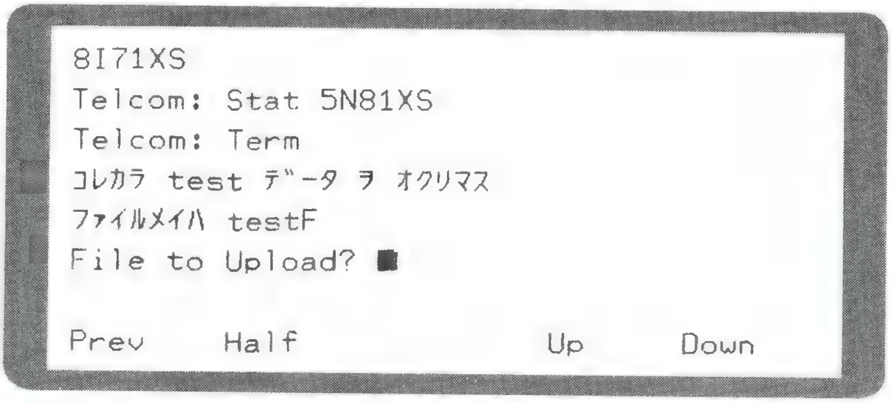
ファイルの内容がすべて送り出されると，送信は終わり，反転していた「Up」の文字は元へ戻り，再びTERMモード，つまりキーボードから入力できる状態になります。ファイルの送信中はキー入力は行っても無視されますが，送信はできます。HALF／FULLの指定にかかわらず，PC-8201は全二重通信（送受信が同時に可能な通信方式）だからです。

DOWNLOAD

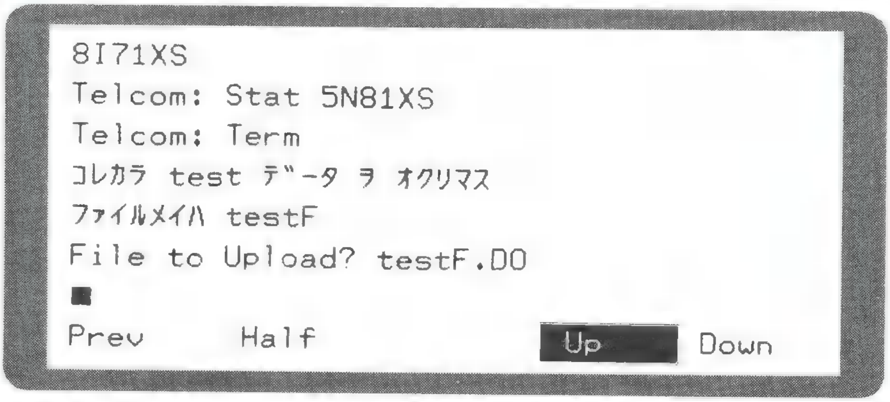
TERMモードでファンクションキーの5番に「Down」と登録されているコマンドで，RS-232C回線からのデータをRAM上にファイルとして記憶します。

[f･5]を押すと，PC-8201は，実行中のTERMモードを一時中断して，RAM上に記録する時のファイル名を，「File to Download?」ときいてきます（図258）。

【図256】



【図257】





受信したデータは，TEXT ファイルとして記録されるため，指定できるのは TEXT ファイル（.DO 付き）だけです。拡張子 .DO は省略できます。

ファイル名が正しく入力されると，［⏎］キーを押した直後から，受信したデータをすべてファイルへ書き込みます。受信中は，「Down」の文字が反転しています(図259)。

受信データは，ファイルへ記録されると同時に画面にも表示されていますから，受信内容を知ることもできます。

DOWNLOAD の状態は自動的には終わりません。「Down」の文字の反転はそのままです。必要なデータがファイルに記録された

【図258】

```
Telcom: Stat
5N81XS
Telcom: Term
File to Download? ■




Prev     Half                 Up       Down
```

【図259】

```
Telcom: Stat
5N81XS
Telcom: Term
File to Download? ZAIKO
■



Prev     Half                 Up       Down
```

【図260】

```
199,PC-8201,ﾊﾝﾄﾞﾍﾙﾄﾞｺﾝﾋﾟｭｰﾀ,48
200,PC-8201-06,ｿﾞｳｾﾂRAM,100
201,PC-8271-01,ACｱﾀﾞﾌﾟﾀ,50
202,PC-8293,CMTｹｰﾌﾞﾙ,50
203,PC-8294,ﾌﾟﾘﾝﾀｹｰﾌﾞﾙ,35
99999,END
■
 Prev   Half              Up      Down
```

ならば（送信側から終わりの合図を送ってもらって画面で確かめるなど），[f·5] を押すとDOWNLOAD状態が終了し，TERM 状態に戻ります（図260）。

何らかの理由でデータを正しく受信できなかった場合，PC-8201 は受信を中止せず，エラーの発生した文字を市松模様に置き換えて続行します。しかし，途中でメモリが不足した場合は，ブザー音で警告し，失敗の意味の DOWNLOAD ABORTED を表示して TERM モードへ戻ります。

受信側が ABORTED により TERM モードに戻っても ABORTED されたことが伝えられないため，送信側からはそのままデータが送られてきます。また，受信したデータは，ファイルには記録されていませんが，画面上に表示は続けて行われます。

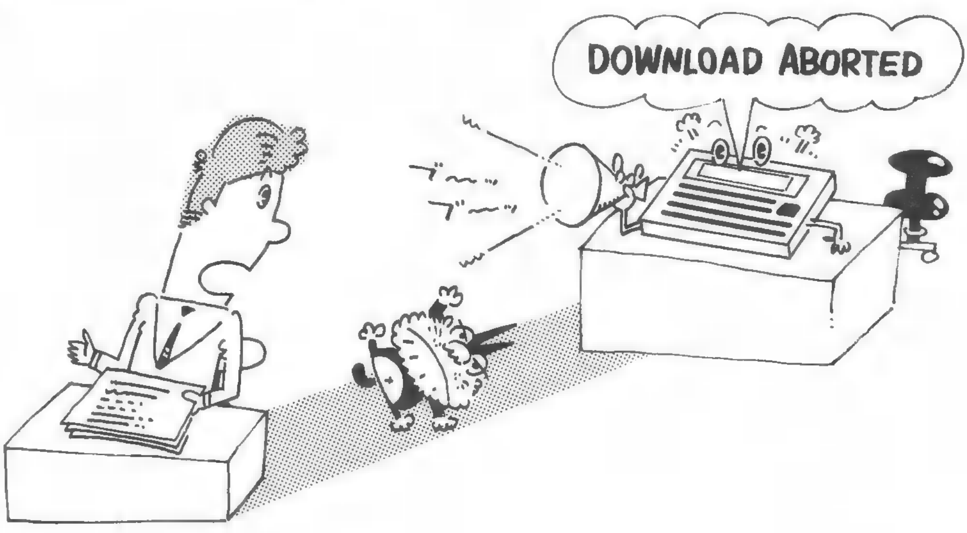
BYE

TERM モードでファンクションキーの10番に登録されているコマンドで，TERM モードから TELCOM モードへ戻る働きをします。

[f·10] を押すと，PC-8201 は，RS-232C 回線の通信を終了し，TELCOM モードへ戻ります。

エスケープシーケンス

PC-8201 では，BASIC モードと同様に，TERM モードにおいても，エスケープシーケンスが使えます。エスケープシーケンスとは，エスケープコード（キーボード上の [ESC] キーを押す，またはキャラクタコードの27）に続く文字により，ある特定の機能



を行わせるものです（表13）。

コントロールコード

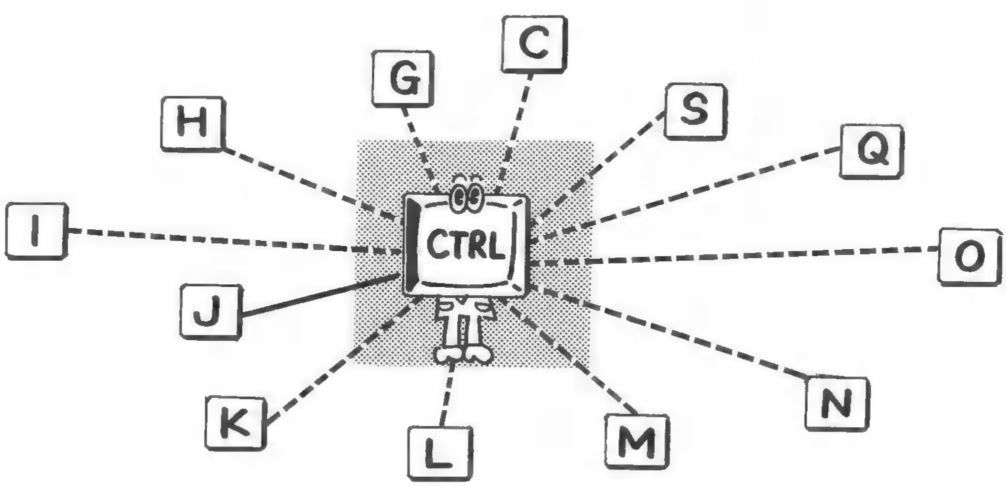
PC-8201では，キャラクタコードの0～31をコントロールコードとして使い，画面操作の機能を持たせています。キー操作上は CTRL キーを押しながら，アルファベットキーを押して，各機能を働かせます。コントロールコードの機能は，各モードごとに異なります。TELCOM におけるコントロールコードは表14のとおりです。

【表13】 TELCOMのエスケープシーケンス

操　作	キャラクタコード	機　　能
ESC + E	27, 69	画面クリア
ESC + j	27,106	画面クリア
ESC + K	27, 75	カーソル位置から行末までの文字を消去
ESC + J	27, 74	カーソル位置から画面の終りまでの文字を消去
ESC + l	27,108	カーソルのある行の文字を消去
ESC + L	27, 76	1行挿入
ESC + M	27, 77	カーソルのある行を削除
ESC + Y <y><x>		カーソルを指定位置へ移動
ESC + A	27, 65	カーソルを1行上へ移動
ESC + B	27, 66	カーソルを1行下へ移動
ESC + C	27, 67	カーソルを1文字右へ移動
ESC + D	27, 68	カーソルを1文字左へ移動
ESC + E	27, 69	カーソルを画面左上隅（ホームポジション）へ移動
ESC + p	27,112	白黒反転文字にする
ESC + q	27,113	文字を正常に戻す
ESC + T	27, 84	ファンクションキー表示を行う
ESC + U	27, 85	ファンクションキー表示を消す
ESC + V	27, 86	スクロール禁止（画面を固定）
ESC + W	27, 87	スクロール許可
ESC + P	27, 80	カーソルを表示する
ESC + Q	27, 81	カーソルを表示しない

【表14】 TELCOMのコントロールコード

操　作	キャラクタコード	機　　能
CTRL + C	3	コマンド入力の中止 （キーボード入力時のみ有効、STOP に同じ）
CTRL + G	7	ベル
CTRL + H	8	バックスペース（DEL BS に同じ）
CTRL + I	9	タブ（TAB に同じ）
CTRL + J	10	ラインフィード
CTRL + K	11	ホームポジション
CTRL + L	12	画面クリア
CTRL + M	13	キャリッジリターン（⏎ に同じ）
CTRL + N	14	シフトアウト（制御指定時のみ有効）
CTRL + O	15	シフトイン（制御指定時のみ有効）
CTRL + Q	17	送信再開許可（制御指定時のみ有効）
CTRL + S	19	送信中断要求（制御指定時のみ有効）
ESC	27	エスケープシーケンスの始まり
←	28	カーソルを1文字右へ移動
→	29	カーソルを1文字左へ移動
↑	30	カーソルを1行上へ移動
↓	31	カーソルを1行下へ移動



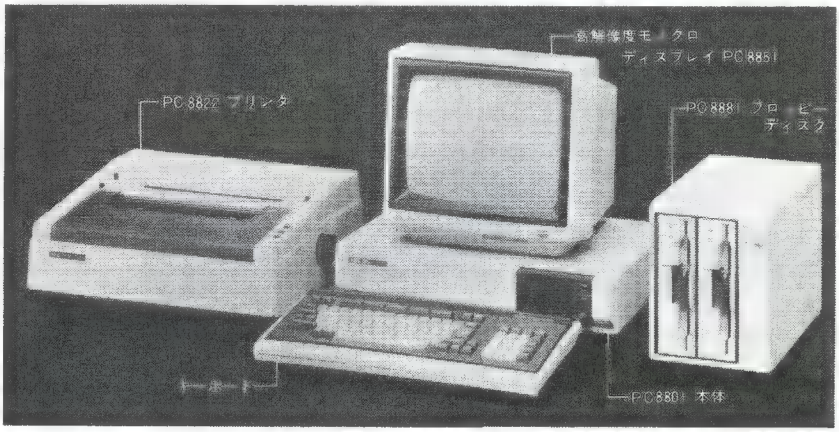
③ TELCOMの使用例

PC-8201の機能が大体わかったところで，実際に他のコンピュータと接続してデータ通信を行ってみましょう。

ここではPC-8201とPC-8801をRS-232Cケーブルで接続して，データの送受信を行います。

他のコンピュータであってもよいのですが，PC-8801側のプログラムを説明する時に，同じBASICを使っているので，わかりやすいと思います。また，電話回線を使って通信する場合は，別に音響カプラを必要とするだけで方法はまったく同じです。

【写真】
PC－8800シリーズの構成



PC-8801との接続

PC-8201とPC-8801は，いずれもRS-232Cインターフェイスを標準装備しています（図261）。

RS-232Cケーブル（ここでは，NECのPC-8895を使いましたが，どのメーカーのものでも同じですし，手作りしてもかまいません）を両方のコネクタピンに差し込みます（図262）。

起　動

2の「TELCOMコマンドの使い方」で説明したとおり，メニュー画面からTELCOMを選択します。画面はクリアされ，設定されている通信形式の値を表示します（図263）。

ここで現在の通信形式のままでよければ，[f·5]を押してTERMモードへ入ります。変更したい場合は，[f·4]を押しSTATの文字を表示させ，続いて通信形式を入力します。

いま，5N81XSを入力してください。この通信形式の意味は，

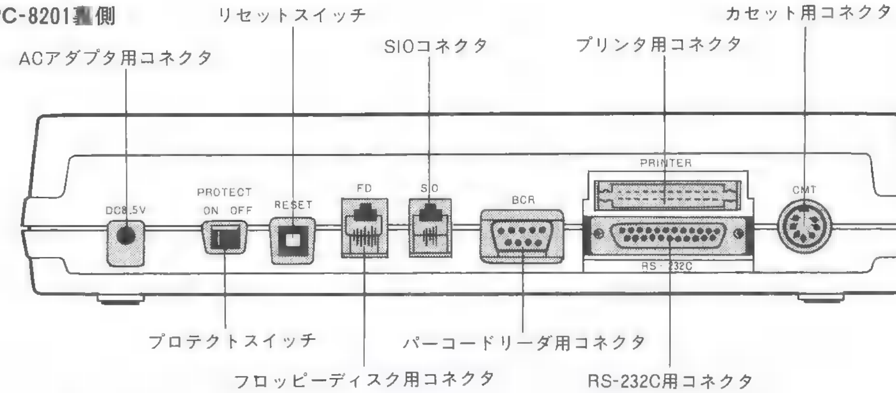
1200bpsの通信速度
パリティビットはなし
文字は8ビット長
文字の区切りに1ビット
Xパラメータ制御を行う
SI／SO制御を行う

です（図264）。

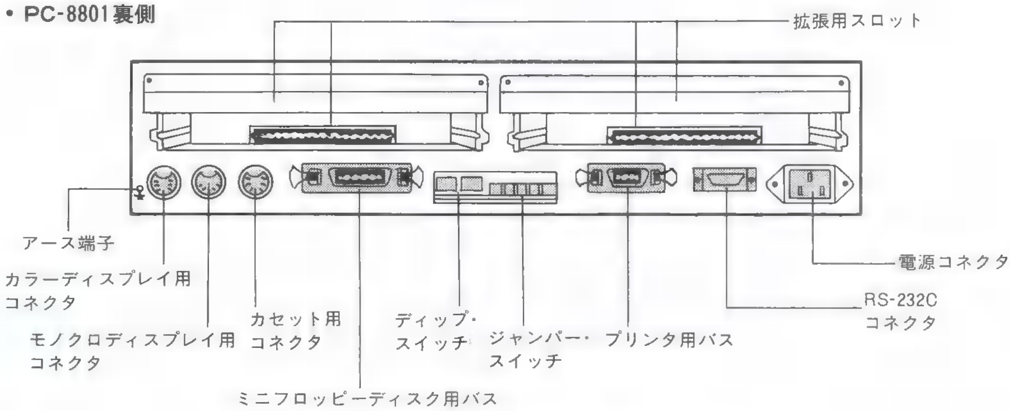
通信形式は別のものを設定してもかまいま

【図261】

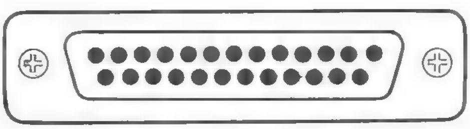
• PC-8201裏側

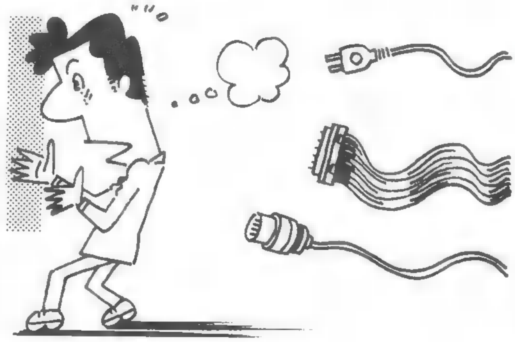


• PC-8801裏側

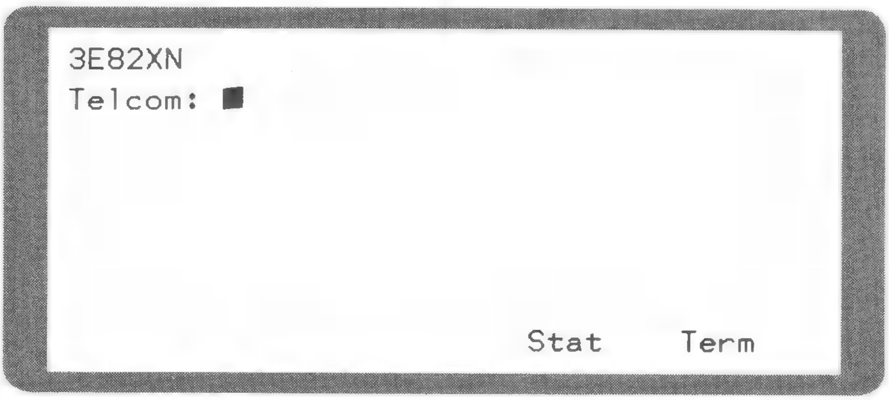


【図262】RS-232Cインターフェイスのコネクタ

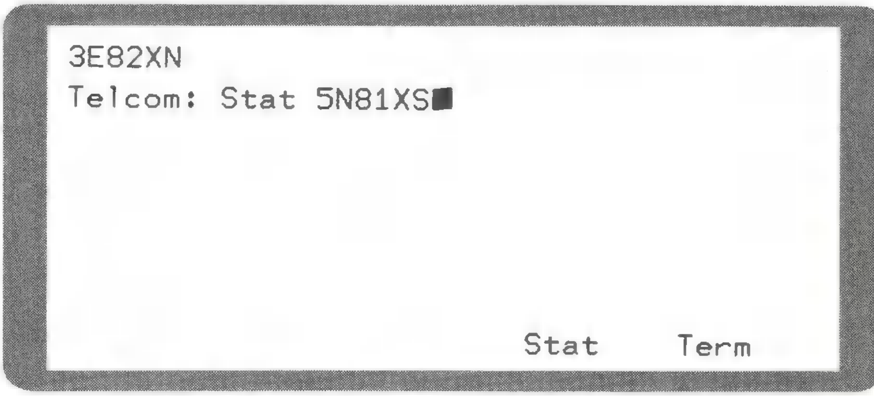




【図263】



【図264】



せんが，両方のコンピュータで同じ通信形式の設定を行う必要があります。また，これ以上高速の処理をしても，PC-8201 の性能から無理があります。

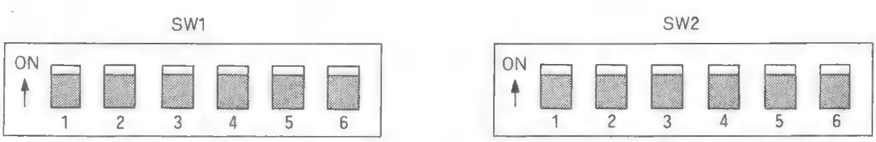
通信形式の指定の次に，[f·5] を押せば TERM モードに入ります。このあとは，PC-8801 側を準備し，キー入力によりデータを送るか，UPLOAD または，DOWNLOAD の選択により，ファイルの転送を行うことになります。

PC-8801側の準備

PC-8801で，RS-232Cインターフェイスを使ってデータの送受信を行う方法には，2通りの方法があります。

1つは，TERM コマンドを実行して，PC-8801 をターミナルモードにして，より上位のホストコンピュータのキーボードターミナルとして使う場合で，もう1つは N_{88}-BASIC 内で，RS-232C ポートをファイル

【図265】 ディップ・スイッチ



SW1	1	ON	N-BASIC
		OFF	N_{88}-BASIC
	2	ON	ターミナルモード
		OFF	BASIC
	3	ON	80文字／行
		OFF	40文字／行
	4	ON	25行／画面
		OFF	20行／画面
	5	ON	Sパラメータ有効
		OFF	Sパラメータ無効
	6	ON	DELコードを処理する
		OFF	DELコードを無視する

SW2	1	ON	パリティ有り
		OFF	パリティ無し
	2	ON	偶数(EVEN)パリティ
		OFF	奇数(ODD)パリティ
	3	ON	8ビット
		OFF	7ビット
	4	ON	ストップ・ビット＝2
		OFF	ストップ・ビット＝1
	5	ON	Xパラメータ有効
		OFF	Xパラメータ無効
	6	ON	半二重
		OFF	全二重

の一つとして扱う方法です。PC-8201 と接続して使う場合は，２番目の N_{88}-BASIC による制御の方が適しています。PC-8801 および N_{88}-BASIC についての詳しい説明が必要な方は，PC-8801 のユーザーズマニュアルとリファレンスマニュアルを参照してください。

PC-8801の裏側には，12個のディップ・スイッチと17個のジャンパー・スイッチがあります（図265，図266）。

PC-8801 をターミナルモードで使う場合は，これらのスイッチによって，通信形式の設定をハードウェア的に行いますが（ソフトウェア的にも可能です），BASIC で制御する場合は，ボーレート切り換えのみ行います。

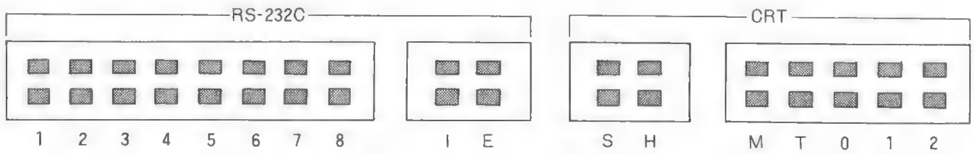
ソケットを PC-8201 の通信速度に合わせて，1200ボー，つまり５のところへ差し込んでください。なお，クロックは直接 PC-8201 と接続する場合は，I に，音響カプラにより電話回線を使って接続する場合は，E にします。いまは，I にソケットを差し込みます。

スイッチの設定が終わったら，PC-8801 およびその周辺装置に電源を入れます（図267）。

ここでは，PC-8801 のディスク付きのシステムを考えていますが，ディスクはなくても，通信にはさしつかえありません。初期画面が出たら，[2] を入力し [⏎] キーを押します（図268）。

ここで，図269のプログラムを入力してく

【図266】 ジャンパー・スイッチ



ボーレート切替(RS-232C)	1	75ボー
	2	150ボー
	3	300ボー
	4	600ボー
	5	1200ボー
	6	2400ボー
	7	4800ボー
	8	9600ボー
クロック(R×C端子)	I	内部クロック
	E	外部クロック
グラフィックでのタテの最大表示ドット数	S	200ドット
	H	400ドット
白黒モニタへの出力画面	M	テキスト＋グラフィック1，2，3
	T	テキスト
	0	グラフィック1
	1	グラフィック2
	2	グラフィック3

【図267】

```
Disk version [Aug 20,1982]
How many files(0-15)? ■
```

【図268】

```
Disk version [Aug 20,1982]
How many files(0-15)? 2
NEC N-88 BASIC Version 1.1
Copyright (C) 1981 by Microsoft
45410 Bytes free
```

ださい。もちろん，あらかじめ入力してフロッピーディスクへ記憶してあれば，ロードするだけですみます。

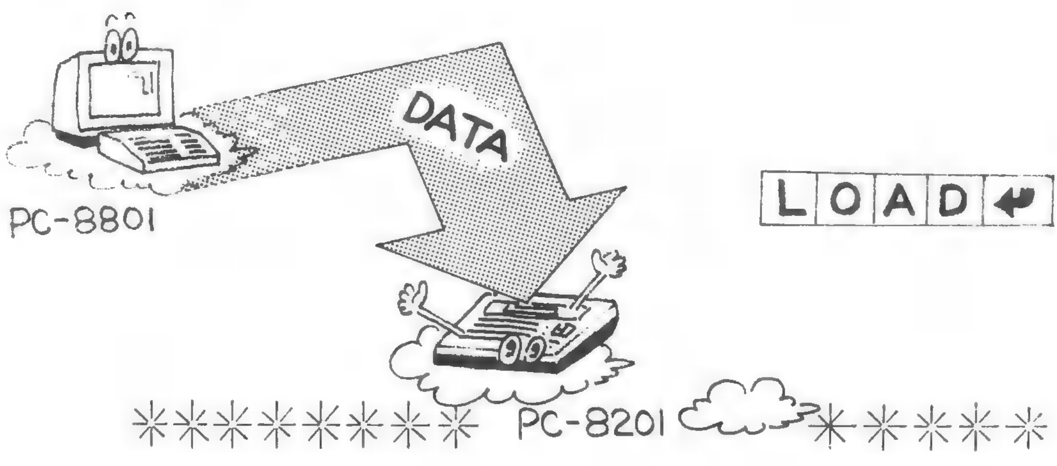
このプログラムで，N_{82}-BASIC と異なる命令もいくつかあります。行番号10，90，150，220，360の画面制御命令は削除しても，全体の流れには影響しません。画面をみやすくしているものなので，削除した場合はLOCATE の座標を変える必要があります。

行番号100をみてください。RS-232C ポートを１つのファイルとみなして，OPEN 命令で開いています。OPEN の後にはファイル名ではなく，RS-232C ポートであることを表す，装置名 COM：を書き，続けて通信形式を５桁で定義します。

"COM：N 81 XS"

ここでは，PC-8201 側の通信形式の設定 5 N 81 XS と同じに設定しています。ボーレートの方は，PC-8801 のジャンパースイッチで 1200 bps を指定しているため，プログラム内では指定しません。RS-232C ポートからは，入力・出力の両方が可能ですから，特に FOR INPUT，FOR OUTPUT を指定しません。OPEN 後は，通常のシーケンシャルファイルの処理とまったく同じように扱います。

PC-8201 からのデータは，１ビット単位に分解され，シリアルに送られてきますが，文字の分解および復元は RS-232C が行いますから，プログラムでは，LINE INPUT # で ⏎ キーまでのデータを入力し，PRINT # で出力します。



【図269】

```
10 CONSOLE 0,25,0,0:WIDTH 80,25:COLOR 0
20 SCREEN 0,0:CLS
30 LOCATE 8,0:PRINT "1.  PC-8801 ｷｰﾎﾞｰﾄﾞ TO PC-8201
40 LOCATE 8,1:PRINT "2.  PC-8801 ﾌｧｲﾙ    TO PC-8201
50 LOCATE 8,2:PRINT "3.  PC-8201     TO PC-8801
60 LOCATE 8,3:PRINT "4.  PC-8201     TO PC-8801 ﾌｧｲﾙ
70 LOCATE 10,4:INPUT "ｼｮﾘ ｦ ﾊﾞﾝｺﾞｳ ﾃﾞ ｲﾝﾌﾟｯﾄ";A%
80 IF A%<1 OR A%> 4 THEN GOTO 20
90 CONSOLE 14,8
100 OPEN "COM:N81XS" AS #1
110 ON A% GOTO 130,170,270,270
120 GOTO 20
130 *SEND1
140 LINE INPUT A$
150 IF A$="999" THEN PRINT #1,A$:CLOSE:CONSOLE 0,25,1:END
160 PRINT #1,A$:GOTO 130
170 *SEND2
180 LOCATE 0,4:FILES
190 LOCATE 8,22:INPUT "FILE NAME";F$
200 LOCATE 0,4:FOR I= 1 TO 10:PRINT SPACE$(80);:NEXT:LOCATE 0,14
210 OPEN F$ FOR INPUT AS #2
220 IF EOF(2) THEN PRINT #1,"999":CLOSE:CONSOLE 0,25,1,0:END
230 LINE INPUT #2,A$
240 PRINT #1,A$
250 GOSUB *BEEP5:PRINT A$
260 GOTO 220
270 *RECEIVE
280 IF A%=3 THEN GOTO 330
290 LOCATE 0,4:FILES
300 LOCATE 8,22:INPUT "FILE NAME";F$
310 LOCATE 0,4:FOR I= 1 TO 10:PRINT SPACE$(80);:NEXT:LOCATE 0,14
320 OPEN F$ FOR OUTPUT AS #2
330 LINE INPUT #1,A$:GOSUB *BEEP5:PRINT A$
340 IF A%=3 THEN 360
350 PRINT #2,A$
360 IF A$="999" THEN CLOSE:CONSOLE 0,25,1:END
370 GOTO 330
380 *BEEP5
390 BEEP 1:FOR I= 1 TO 30:NEXT:BEEP 0
400 RETURN
```



操　作

PC-8201 側は TELCOM を，PC-8801 側を図269のプログラムを使って，PC-8201 からデータを送り，PC-8801 側のフロッピーディスクへ記録します。

PC-8801 で図269を RUN させます。メニュー画面が表示されますので，[4]を入力します（図270）。

現在フロッピーに入っているファイル名を表示し，新しく作るファイルの名前をきいてきます(図271)。test F と入力します(図272)。PC-8801 は，test F を OPEN し，RS-232C からの入力待ちになります。

PC-8801 の準備ができた後，PC-8201 からデータ送信を開始します。現在 TERM モ

【図270】

```
1.  PC-8801 ｷｰﾎﾞｰﾄﾞ TO PC-8201
2.  PC-8801 ﾌｧｲﾙ    TO PC 8201
3.  PC-8201      TO PC-8801
4.  PC-8201      TO PC-8801 ﾌｧｲﾙ
  ｼｮﾘ ｦ ﾊﾞﾝｺﾞｳ ﾃﾞ ｲﾝﾌﾟｯﾄ? ■
```

【図271】

```
        1.  PC-8801 ｷｰﾎﾞｰﾄﾞ TO PC-8201
        2.  PC-8801 ﾌｧｲﾙ    TO PC-8201
        3.  PC-8201      TO PC-8801
        4.  PC-8201      TO PC-8801 ﾌｧｲﾙ
backup.n88 2    format.n88 2    setinf.n88 1    xfiles.n88 1    EX-4   .    1
EX-3   .    1   EX-2   .    1   EX-1   .    1   EX-33       1   game1  .    1
game3  .    1   game2  .    1   ONLINE.     1   testf       1   filepr.int 1
game4

        FILE NAME? ■
```

【図272】

```
1.  PC-8801 ｷｰﾎﾞｰﾄﾞ TO PC-8201
2.  PC-8801 ﾌｧｲﾙ    TO PC-8201
3.  PC-8201      TO PC-8801
4.  PC-8201      TO PC-8801 ﾌｧｲﾙ

FILE NAME? testF
```

ードであれば，キー入力したものはすべてそのまま送信されます。入力が終わったら，その旨を受信側に伝えるために，[9] [9] [9] を入力します。

キー入力したものをそのまま送信してもよいのですが，入力ミスなどもありえますから，あらかじめ TEXT ファイルとして RAM 上に記録されているものを送信した方が効率が上がります。

ファイルを転送する時は，PC-8201 の TERM モードで，[f·4] を押します。送信すべきファイルの名前を聞いてきますのでここでは test F を入力します。RAM 上にないファイル名および TEXT ファイル以外を指定すると，ファイルがみつからない（NO FILE）と表示し，ABORT されます。

ファイル名を入力し，[⏎] を押すと送信がはじまります。送信中は,「Up」の文字が反転しています（図273)。

PC-8201 では，送信内容は，HALF に設定しているため，すべて表示されます。PC-8801 側では，受信したデータを表示するように図269でプログラムしてありますから，こちらも，すべて表示します。

送信側はファイルの終わり（EOF）に達したら，自動的に終わることができますが，受信側は，そのままでは終わりません。いまの例の場合，EOF に達すると,「Up」の反転は平常に戻り，また，TERM モードになりますが，PC-8801 の方は，続けて受信待ち

【図273】

```
5N81XS
Telcom: Term
File to Upload? testF
■



 Prev   Half              Up     Down
```



状態です。そのため PC-8201 側から終わりの印，[9] [I] [9] を送ります。PC-8801 側のプログラムは，[9] [9] [9] を受信した時に RS-232C ポートを CLOSE し，プログラムを終了するように作ってあります。

PC-8801 から，PC-8201 へ送信する場合も同様の方法で行います。先に受信側である PC-8201 を起動し，TERM モードにします。受信した内容はすべて画面表示されますが，ファイルに記録するのであれば，さらに [f·5] を押して，ファイル名を入力しておきます。印刷が必要であれば，[f·3] を押しておきます。ここでは，PC-8801 側のフロッピーからプログラムを送信し，PC-8201 で印刷しながら，たとえば EX-61 の名で記録します（図274）。

次に PC-8801 を起動させ，図269を実行させます。メニューから，2 の処理を選択します。FILES 命令により，フロッピーディスクの内容が表示されるので，送るべきファイル名を選択します。

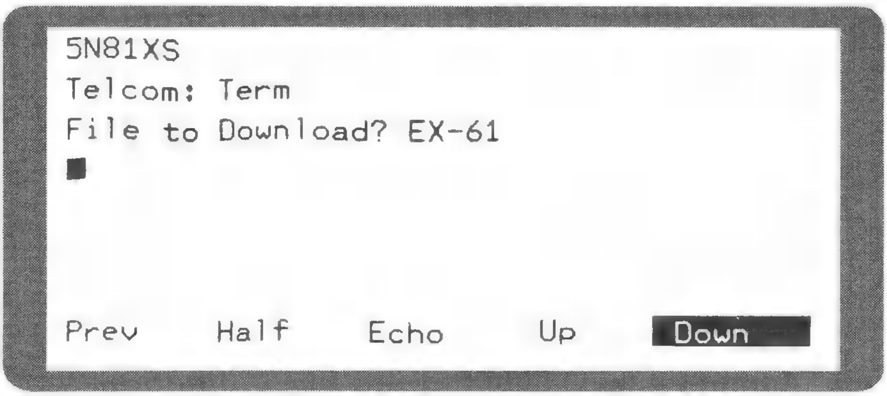
PC-8801 では，TEXT ファイル，BASIC ファイルの区別はありませんから，フロッピー上のどんなファイルでも送信することができます。

しかし，N_{82}-BASIC と N_{88}-BASICでは，BASIC の内部コードがまったく違うため，N_{88}-BASIC で書かれたプログラムをそのまま送っても，PC-8201 側ではそれを使うことができません。

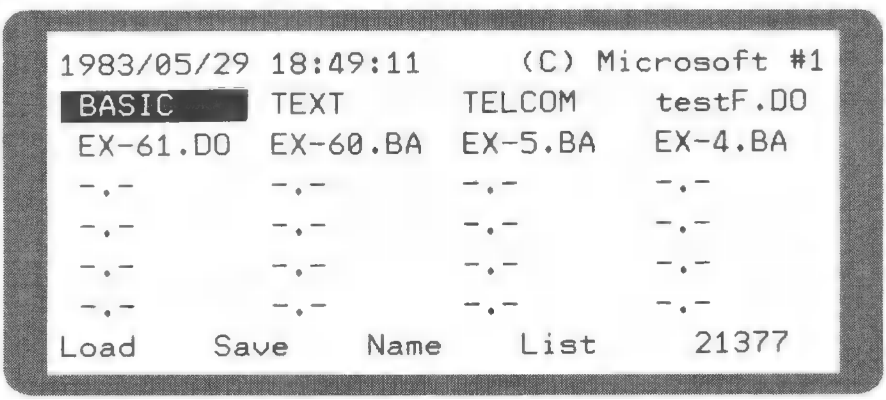
そのため，PC-8801 側からプログラムを送信する場合は，あらかじめアスキーセーブしたものに限ります。いま，図269をアスキーセーブしたもののファイル名を EX-100 として，[E] [X] [−] [1] [0] [0] [⏎] と入力すると送信がはじまります。

EX-100 の内容がすべて送信されると，図269のプログラムは終わりの印として [9]

【図274】



【図275】



9 9 を送り，プログラムは終了します。PC-8201 の画面に999が表示されたら，f·5 を押して，DOWN LOAD を終わらせます。

ここで，f·10 を押します。TERM モードから TELCOM モードへ戻ります。さらにもう一度 f·10 を押します。TELCOM モードから，メニューモードへ戻ります（図275）。

EX-61 の名の TEXT ファイルが作成されています。TEXT ファイルであっても，内容はアスキーセーブしたプログラムですから，BASIC モードでロードすることはもちろんできます。

ロード後，あらためて BASIC ファイルとしてセーブすれば，RAM 上に，.BA 付きのファイルとして記録され，これ以降 PC-8201 でこのプログラムを使うことができるようになります。

EX-61 には，N_{82}-BASIC で使えない命令がいくつか含まれているため，実際には，このまま実行すると？SN ERROR となります。PC-8801 側で PC-8201 用に修正してから送るか，PC-8201 側で修正するかの処置が必要です。このプログラムを PC-8201 用へ修正した例は，図276を参照してください。

TELCOMを使わない送信

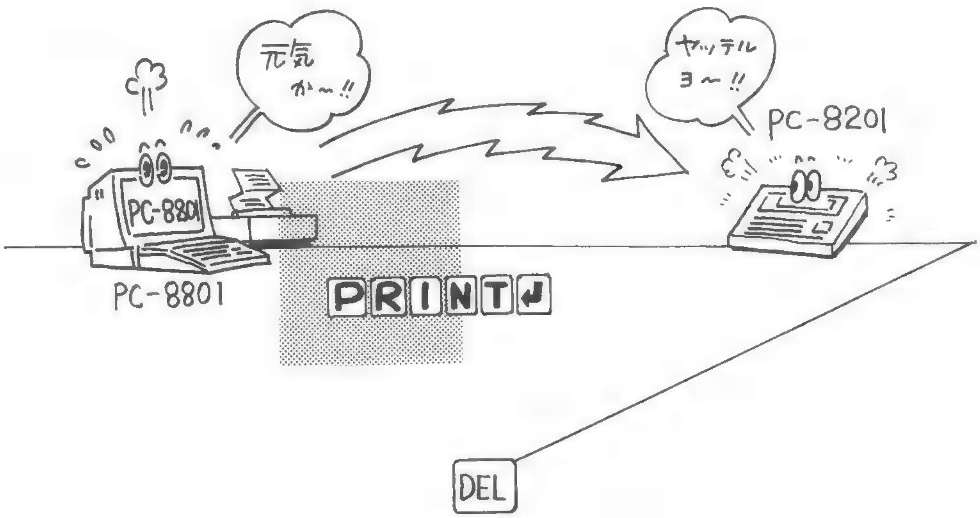
PC-8201 には，TELCOM という強力な通信用ソフトウェアがありますから，たいていの通信は TELCOM で十分と思います。しかし，他のコンピュータからの通信を待っている間，別の仕事をさせたいとか，送信の時にも印刷したいとかの希望がある場合は，TELCOM では対処できません。そこで，N_{82}-BASIC でプログラムを作ることになります。

図276のプログラムは，N_{82}-BASIC による通信用プログラムです。

図276は，図269 をベースに作成したものです。したがって，新たにキーボードから入力せずに，EX-61 を修正した方が簡単です。

COM を OPEN する部分以外は，特に新しい命令はありません。

もう一つ図277の例をみてください。これは，メインの処理をしている間に，RS-232 C ポートへデータが送られてきた場合に，現在の仕事に割り込んで受信されたデータを処



理し，それが終わったら，また元のメインの処理へ戻るプログラムです。この割り込みに関する命令が，ON COM GOSUB と COM ON です。

ON COM GOSUB

この命令は，プログラム内で RS-232 C ポートからの割り込みが発生した時に分岐するサブルーチンの行番号をあらかじめ宣言しておくものです。この命令がプログラム内に存在しても，実際に割り込みが発生しなかったり，COM ON の命令が実行されていなければ，このサブルーチンへ分岐することはありません。

COM ON

ON COM GOSUB 命令で割り込みを認めている場合でも，プログラムのある部分では，割り込みを停止したい場合があります。RS-232 C ポートからの割り込みを許可するのがCOM ON，停止するのがCOM STOPです。

【図276】

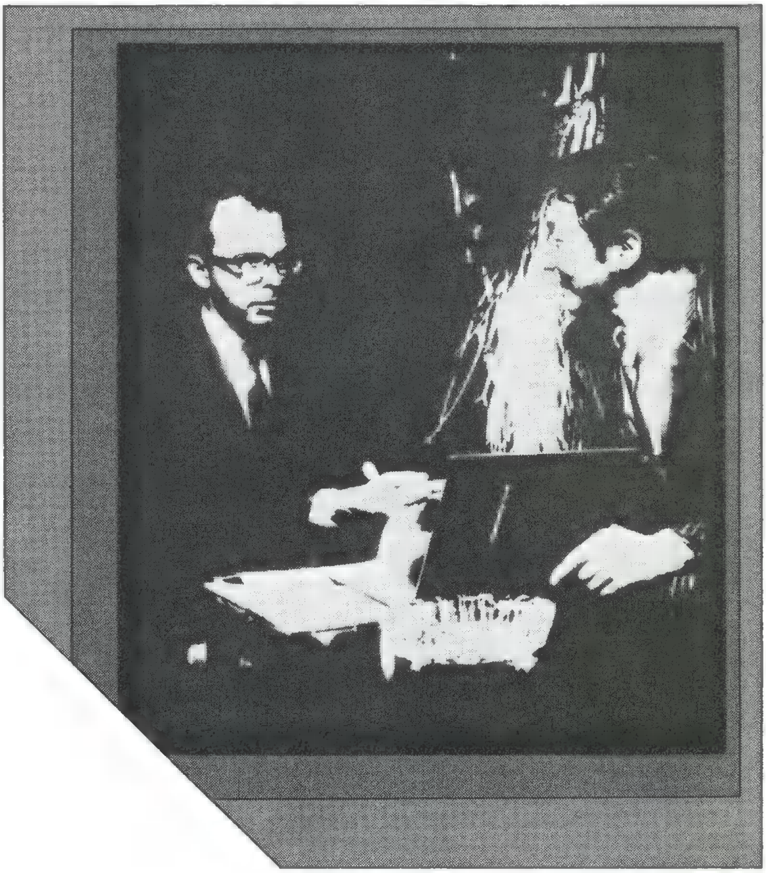
```
10 MAXFILES=2
20 SCREEN 0,0:CLS
30 LOCATE 8,0:PRINT "1. SEND FROM ｷｰﾎﾞｰﾄﾞ"
40 LOCATE 8,1:PRINT "2. SEND FROM ﾌｧｲﾙ"
50 LOCATE 8,2:PRINT "3. RECEIVE TO ﾃﾞｨｽﾌﾟﾚｲ"
60 LOCATE 8,3:PRINT "4. RECEIVE TO ﾌｧｲﾙ"
70 LOCATE 10,4:INPUT "ｼｮﾘ ｦ ﾊﾞﾝｺﾞｳ ﾃﾞ ｲﾝﾌﾟｯﾄ";A%
80 IF A%<1 OR A%>4 THEN GOTO 20
90 ON A% GOTO 120,170,290,260
100 GOTO 20
110 '
120 'SEND1
130 OPEN "COM:5N81XS" FOR OUTPUT AS #1
140 LINE INPUT A$
150 IF A$="999" THEN PRINT #1,A$:CLOSE:END
160 PRINT #1,A$:GOTO 140
170 'SEND2
180 OPEN "COM:5N81XS" FOR OUTPUT AS #1
190 LOCATE 8,5:INPUT "FILE NAME";F$
200 OPEN F$ FOR INPUT AS #2
210 IF EOF(2) THEN PRINT #1,"999":CLOSE:END
220 LINE INPUT #2,A$
230 PRINT #1,A$
240 BEEP:PRINT A$
250 GOTO 210
260 'RECEIVE
270 LOCATE 8,5:INPUT "FILE NAME";F$
280 OPEN F$ FOR OUTPUT AS #2
290 OPEN "COM:5N81XS" FOR INPUT AS #1
300 LINE INPUT #1,A$:BEEP:PRINT A$
310 IF A$="999" THEN CLOSE:END
320 IF A%=3 THEN 300
330 PRINT #2,A$
340 GOTO 300
```

【図277】

```
10 MAXFILES=2
20 OPEN "com:5N81XS" FOR INPUT AS #1
30 SCREEN 0,0:CLS
40 ON COM GOSUB 120
50 COM ON
60 ' MAIN ROUTINE
70 FOR X= 1 TO 255
80  FOR Y= 2 TO 255
90 PRINT X;Y;
100 NEXT Y,X
110 GOTO 60
120 'RECEIVE
130 LINE INPUT #1,A$:BEEP:PRINT A$
140 IF A$="999" THEN CLOSE #1:RETURN
150 LPRINT A$
160 GOTO 130
```

第5章

PC-8201の活用



●PC-8201 の活用

これまで説明してきましたように，PC-8201 は，RAM 上にファイルが作成できることと，TEXT, TELCOM のソフトウェアを内蔵しているため，非常に使いやすいパソコンです。

ハンドヘルドとしての実際の使用例では，金融機関の外回りの営業マンが即座に何種類もの金利計算を行い，商談に役立たせるとか，不動産の同じく営業マンが商談しながら，電話回線を通じて本社のコンピュータが管理している物件を顧客に提示するなどがあげられます。小型のプリンタも一緒にアタッシュケースに入れて持ち運び，その場で表や設計図の印刷も可能です。

また，電卓感覚で一人ひとりがオフィスや書斎の机の引き出しに入れておき，必要な時に取り出して使うということも増えてきています。この電卓感覚で使うためのプログラムとして，PC-8201 用に開発された簡易ソフトを紹介します。

このプログラムを動かすための機器構成は次のとおりです。

PC-8201（32 K バイト RAM）

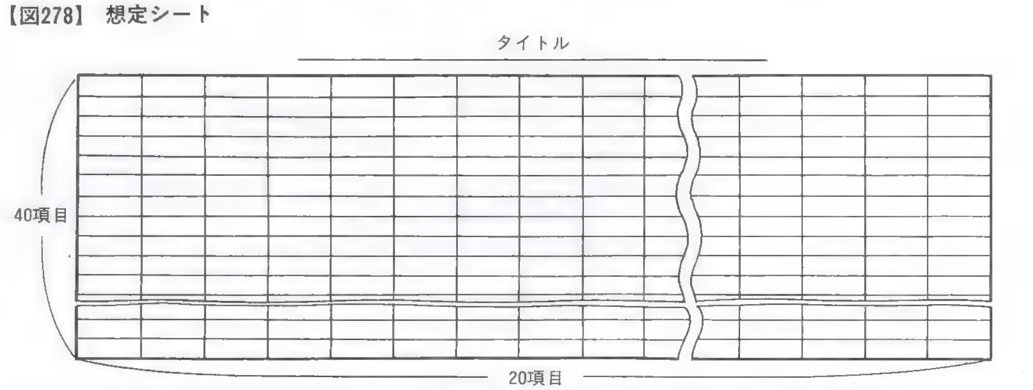
データレコーダ

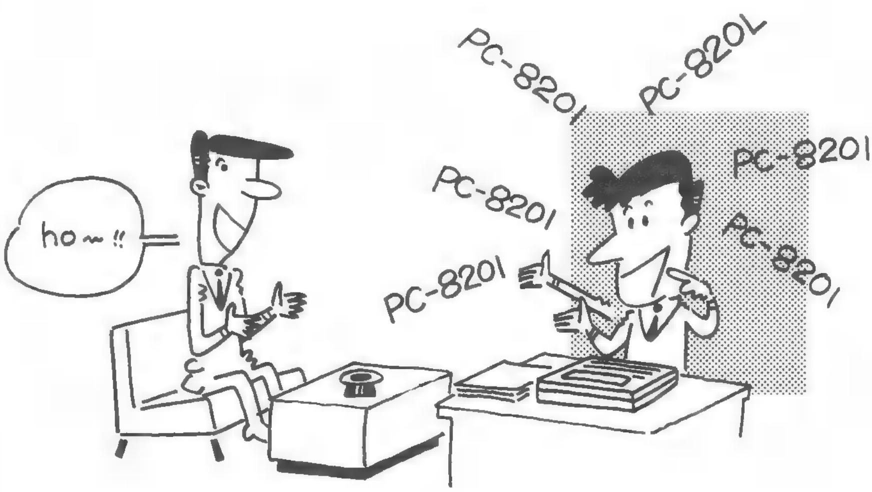
プリンタ

データレコーダとプリンタは，あると処理結果をカセットテープに保存したり，印刷できるということで，なくてもかまいません。ただし，プログラムを毎回タイプするのは大変ですから，プログラムの呼び出し用として，データレコーダは必要です。

日常の仕事の中で，計算を行い表にまとめ

【図278】 想定シート





【表15】 マトリックス計算コマンド

コマンド	機　能	解　　説
ID	データ入力	カーソル部分の項目にデータを入力する。カーソル移動キーを使用すると，どの項目からでも，データの入力ができる(左上のR・L・D・Uはカーソルの動く方向)。 入力中の文字の修正は，DELキーで行う。ESCキーによりコマンドモードに戻る。
IT	タイトル入力	入力中の文字の修正は，DELキーで行う。 タイトル入力後リターンキーにより，コマンドモードに戻る。
IN	項目名入力	入力中の文字の修正は，DELキーで行う。カーソル移動キー　→・←　は使用可能。 ESCキーによりコマンドモードに戻る。
L	罫線出力	指定行列に罫線を出力する。　V　列項目に罫線出力 └── 列番号（A－T） H　行項目に罫線出力 └── 行番号（1－40） ESCキーによりコマンドモードに戻る。
K	計　算	このコマンドが入力された時点で計算が行われ，計算結果が出力される。 V：列優先　列（縦方向）を優先して計算する。 H：行優先　行（横方向）を優先して計算する。
IC	計算式入力	カーソル部分の項目に計算式を入力する。計算は入力された順序で行われる。 〈例〉　A1＋B1＊C1　A1とB1を加算し，それにC1を乗ずる。 項目の表現方法は，列項目番号（A－T）と行番号（1－40）で。〈例〉　A1・B13・T40 使用できる演算子 演算　記号　キーボード入力 加算　＋　＋ 減算　－　－ 乗算　×　＊ 除算　÷　／　※（　）は使用不可能。 関数　書式 合計関数　@TL　@TL（項目番号：項目番号） 平均関数　@AV　@AV（項目番号：項目番号） 条件指定（パラメータ） 整数　&I　計算結果を整数で得る。〈例〉&I：@AV（A1：A5） &I：&D：B1／D5 小数点　&Sn　計算結果を小数点第n位で得る。　1≦n≦5 四捨五入　&D　計算結果を四捨五入して得る。 ※&Iを指定している場合は小数点第1位，&Snを指定している場合は，指定小数点位の次の位数で四捨五入される。
F	フォーマット変更	列項目の長さを変更する。 列番号（A－T）を指定　→　項目の長さ（1－35）を指定。 ESCキーによりコマンドモードに戻る。
D	画面表示	画面への表示を変更する。　ZF；項目の左右寄せをする。 └─ 列番号（A－T）　L；左寄せ　R：右寄せ ZT；タイトルの表示　ON；表示する　OFF；表示しない ZN；項目名の表示 ZA；項目番号の表示 ESCキーによりコマンドモードに戻る。
CF	ファイル処理	データのカセットへの入出力を行う。 S　データのセーブ └─ ファイル名　－項目番号（左上端）　－項目番号（右下端） L　データのロード └─ ファイル名　－　〈ファイル名〉ヨミコミチュウ ESCキーによりコマンドモードに戻る。　SAVE／LOAD実行後はコマンドモードに戻る。
P	印　刷	データをプリンタに出力する。　項目番号（左上端）──項目番号（右下端） ESCキーによりコマンドモードに戻る。　P　実行後はコマンドモードに戻る。
MC	カーソル移動	カーソルおよび画面の表示を移動する。 行番号　指定した行番号が上端に表示される。 列番号　指定した列番号が左端に表示される。 項目番号　指定した項目番号が左上端に表示される。 ESCキーによりコマンドモードに戻る。
C	クリア	データのクリアをする。 G；すべて D；入力された文字データ（記号・罫線）および数値データ N；数値データ A；文字データ K；計算式および計算結果

【図279】

```
R  A      B      C      D      E
1
2
3
4
5
6
>>■   [ID(デ゛ータニュウリョク)/IT(タイトルニュウリョク)]?
```

るというのは非常に多いものです。紹介する簡易ソフト「マトリックス計算」は，この計算・作表を行うプログラムです。

このプログラムでは，メモリ内に横20，縦40の項目を持つ1枚のシートを想定します（**図278**）。各項目の使い方は自由で，長さ，見出し，文字か数字か，キー入力するか，計算結果を求めるかなど，すべて簡単なコマンド（全部で12個）で指定できます（**表15**）。

このプログラムを実行すると，画面に表の左上が表示され，一番下にコマンドが表示されます。コマンドは忘れてもいいように，常に一番下の行に表示されています。⏎キーを押すと，次のコマンドを表示します。最後のコマンド（END）まで進むと，また最初のコマンドから表示します（**図279**）。

カーソルがコマンドの行にある時は，コマンドが入力できます。各コマンドの使用順位は自由なので，最初にタイトルを入力し，次に各項目の長さを指定して，というように固定して考える必要はありません。計算してから，桁数をもう少し多くしても良いし，印刷の直前に項目の見出しを入力してもかまいません。

例えば，コマンドのITを入力し，タイトルの指定をしてみます。カーソルは自動的に画面の一番上に移り，表は全体に下へスクロールします。表のタイトルを入力して⏎キーを押すと，またコマンド入力へ戻ります。

ハンドヘルド用の簡易ソフトといっても，実務に生かせるだけの機能を持っていますので，売上表，成績表，予実績対比表，月別集計表など身近な例で活用していただきたいと思います。

なお，このプログラムは，「マトリックス計算」という商品名で，NECのマイコンショップで取り扱っています。価格は8000円です。

なお，近くで買えない方は下記に申し込んでください。

〒150　東京都渋谷区渋谷2—22—3
渋谷東口ビル11F
株式会社　日本情報研究センター
OA営業部　業務課

宅急便でお送りしますので，送料は着払いとなり，地区によって異なりますが，だいたい500～600円です。

〔**参考文献**〕

NEC　PC-8201　USER'S　MANUAL
NEC　PC-8201　N82-BASIC　REFERENCE MANUAL
NEC　PC-8801　USER'S　MANUAL
NEC　PC-8023C　USER'S　MANUAL

マトリックス計算プログラム

```
1 '*** MATRIX *** 58/02/17
2 CLEAR 3000
5 DIM TN$(20),TN%(20),D$(20,40),TZ%(20),TQ%(20,40),DK$(20,40)
7 SCREEN 0,0
8 GOSUB 59000:CX%=1:CY%=1:CF$="R":TB%=0
10 LX%=1:LY%=1:ZT%=0:ZN%=0:ZA%=1:CLS:GOSUB 1500
20 LX%=1:GOSUB 1400:GY%=1:LY%=1:GOSUB 1700
25 LOCATE 1,0:PRINT CF$;
400 '
500 '*** COMMAND INPUT SUB ***
505 PM%=1
510 GOSUB 2000
520 X%=4:Y%=7:L%=3:E%=1:GOSUB 1000
525 IF P$="" THEN IF PM%=7 THEN 505 ELSE PM%=PM%+1:GOTO 510
540 IF P$="ID" THEN DU%=FRE(""):GOSUB 10000:GOTO 500
550 IF P$="IT" THEN GOSUB 12000:GOTO 500
560 IF P$="IN" THEN GOSUB 14000:GOTO 500
570 IF P$="IC" THEN DU%=FRE(""):GOSUB 16000:GOTO 500
580 IF P$="K" THEN GOSUB 18000:GOTO 500
590 IF P$="F" THEN GOSUB 20000:GOTO 500
600 IF P$="L" THEN GOSUB 22000:GOTO 500
610 IF P$="D" THEN GOSUB 24000:GOTO 500
620 IF P$="CF" THEN GOSUB 26000:GOTO 500
630 IF P$="P" THEN GOSUB 28000:GOTO 500
640 IF P$="MC" THEN GOSUB 30000:GOTO 500
660 IF P$="C" THEN GOSUB 34000:GOTO 500
670 IF P$="END" THEN END
680 GOTO 500
1000 '*** input key board sub ***
1010 P$="":P%=0:ZP$="":ZX%=X%:ZY%=Y%:ZL%=0
1020 IF ZL%=L% THEN 1260
1025 IF GG%=1 THEN LOCATE MX%,MY%:PRINT CHR$(130);
1030 LOCATE ZX%,ZY%:PRINT CHR$(130);
1040 ZP$=INKEY$:IF ZP$="" THEN 1040
1045 IF ZP$>CHR$(27) AND ZP$<CHR$(32) THEN IF E%=2 OR E%=3 THEN IF ZL%=0 THEN P%
=2:P$=ZP$:RETURN ELSE 1030 ELSE 1030
1050 IF ZP$=CHR$(127) OR ZP$=CHR$(8) THEN 1320
1060 IF ZP$=CHR$(27) THEN IF E%=3 OR E%=1 THEN IF ZL%=0 THEN P%=3:RETURN ELSE 10
30 ELSE LOCATE X%,Y%:PRINT STRING$(L%," ");:GOTO 1000
1070 IF ZP$=CHR$(13) THEN 1130
1080 IF ZP$<" " THEN 1040
1090 IF ZP$>"ﾟ"THEN 1040
1100 IF E%=1 THEN IF ZP$=";" OR ZP$="," THEN 1040
1110 LOCATE ZX%,ZY%:PRINT ZP$;:P$=P$+ZP$
1120 ZL%=ZL%+1:ZX%=ZX%+1:GOTO 1020
1130 '
1140 IF L%<>ZL% THEN LOCATE ZX%,ZY%:PRINT " ";
1150 ZN%=0:IF P$="" THEN RETURN
1160 FOR ZI%=1 TO LEN(P$)
1170 ZZ$=MID$(P$,ZI%,1)
1180 IF ZZ$="-" OR ZZ$="+" THEN IF ZI%<>1 THEN 1250 ELSE IF LEN(P$)<>1 THEN 1220
 ELSE 1250
1190 IF ZZ$="." THEN IF ZN%=1 THEN RETURN ELSE ZN%=1:GOTO 1220
1200 IF ZZ$<"0" THEN RETURN
1210 IF ZZ$>"9" THEN RETURN
1220 NEXT ZI%
1230 P%=1
1250 RETURN
1260 '
1270 ZP$=INKEY$:IF ZP$="" THEN 1270
1280 IF ZP$=CHR$(13) THEN 1130
1290 IF ZP$=CHR$(8) OR ZP$=CHR$(127) THEN 1350
1300 IF ZP$=CHR$(27) THEN LOCATE X%,Y%:PRINT STRING$(L%," ");:GOTO 1000
1310 GOTO 1270
1320 '
1330 IF ZL%=0 THEN 1030
1340 LOCATE ZX%,ZY%:PRINT " ";
1350 ZX%=ZX%-1:ZL%=ZL%-1
1360 P$=LEFT$(P$,LEN(P$)-1):GOTO 1030
1400 '*** DISPLAY SIZE CALC SUB ***
1405 ZS%=0:RY%=LY%+6-GY%
```

```
1410 FOR ZI%=LX% TO 20
1420 ZS%=ZS%+TN%(ZI%)
1430 IF ZS%>37 THEN RX%=ZI%-1:RETURN
1440 NEXT ZI%
1445 RX%=20
1450 RETURN
1500 '*** TITLE DISPLAY SUB ***
1510 GY%=0
1520 IF ZT%=1 THEN LOCATE 3,0:PRINT TI$;:GY%=1
1530 IF ZA%=0 THEN 1600
1540 GOSUB 1400:LOCATE 0,GY%:PRINT STRING$(40," ");:LOCATE 3,GY%
1550 FOR ZI%=LX% TO RX%
1560 ZN$=STRING$(INT(TN%(ZI%)/2)," "):ZN$=ZN$+CHR$(64+ZI%)
1562 ZN$=ZN$+STRING$(TN%(ZI%)-LEN(ZN$)," ")
1564 PRINT ZN$;
1570 NEXT ZI%
1580 GY%=GY%+1
1600 '
1610 IF ZQ%=0 THEN 1670
1620 GOSUB 1400:LOCATE 0,GY%:PRINT STRING$(40," ");:LOCATE 3,GY%
1630 FOR ZI%=LX% TO RX%
1640 IF LEN(TN$(ZI%))>TN%(ZI%) THEN PRINT LEFT$(TN$(ZI%),TN%(ZI%)); ELSE IF TZ%(
ZI%)=1 THEN PRINT TN$(ZI%);STRING$(TN%(ZI%)-LEN(TN$(ZI%))," "); ELSE PRINT STRIN
G$(TN%(ZI%)-LEN(TN$(ZI%))," ");TN$(ZI%);
1650 NEXT ZI%
1660 GY%=GY%+1
1670 LOCATE 1,0:PRINT CF$;:RETURN
1700 '*** LCD DATA DISPLAY SUB ***
1710 ZX%=LX%:ZZ%=GY%:YY%=LY%
1720 GOSUB 1400:ZP%=RX%
1725 FOR ZJ%=ZZ% TO 6:LOCATE 3,ZJ%:PRINT STRING$(37," ");
1727 LOCATE 0,ZJ%:PRINT RIGHT$(STR$(YY%),LEN(STR$(YY%))-1);" ";:LOCATE 3,ZJ%
1730 FOR ZI%=ZX% TO ZP%
1735 DD$=D$(ZI%,YY%):DL%=LEN(DD$)
1740 IF TB%=1 THEN IF TQ%(ZI%,YY%)=9 THEN IF TN%(ZI%)>LEN(DK$(ZI%,YY%)) THEN PRI
NT STRING$(TN%(ZI%)-LEN(DK$(ZI%,YY%))," ");DK$(ZI%,YY%);:GOTO 1750 ELSE PRINT LE
FT$(DK$(ZI%,YY%),TN%(ZI%));:GOTO 1750
1745 IF DL%>=TN%(ZI%) THEN PRINT LEFT$(DD$,TN%(ZI%)); ELSE IF TZ%(ZI%)=1 THEN PR
INT DD$;STRING$(TN%(ZI%)-DL%," "); ELSE PRINT STRING$(TN%(ZI%)-DL%," ");DD$;
1750 NEXT ZI%:YY%=YY%+1
1760 NEXT ZJ%
1770 RETURN
2000 '*** COMMAND MESSAGE OUTPUT SUB ***
2005 LOCATE 0,7:PRINT STRING$(39," ");
2010 LOCATE 2,7:PRINT ">>";
2020 LOCATE 7,7
2030 IF PM%=1 THEN PRINT "｢ID(ﾃﾞｰﾀﾆｭｳﾘｮｸ)/IT(ﾀｲﾄﾙﾆｭｳﾘｮｸ)｣?";
2040 IF PM%=2 THEN PRINT "｢IN(ｺｳﾓｸﾆｭｳﾘｮｸ)/IC(ｹｲｻﾝﾆｭｳﾘｮｸ)｣?";
2050 IF PM%=3 THEN PRINT "｢K(ｹｲｻﾝ)/F(ﾌｫｰﾏｯﾄﾍﾝｺｳ)｣?";
2060 IF PM%=4 THEN PRINT "｢L(ｹｲｾﾝｼｭﾂﾘｮｸ)/D(LCD ﾋｮｳｼﾞ)｣?";
2070 IF PM%=5 THEN PRINT "｢CF(ﾌｧｲﾙｼｮﾘ)/P(ﾌﾟﾘﾝﾄ)｣?";
2080 IF PM%=6 THEN PRINT "｢MC(ｶｰｿﾙｲﾄﾞｳ)/C(ｸﾘｱ)｣?";
2090 IF PM%=7 THEN PRINT "｢END(ｼｭｳﾘｮｳ)｣";
3000 IF PM%=30 THEN PRINT "｢ZF(ﾐｷﾞ､ﾋﾀﾞﾘﾖｾ)/ZT(ﾀｲﾄﾙ)｣?";
3010 IF PM%=31 THEN PRINT "｢ZN(ｺｳﾓｸﾒｲ)/ZA(ｺｳﾓｸﾊﾞﾝｺﾞｳ)｣";
3020 IF PM%=32 THEN PRINT "｢ﾚﾂﾊﾞﾝｺﾞｳ (A-T)｣";
3030 IF PM%=33 THEN PRINT "｢L(ﾋﾀﾞﾘ)/R(ﾐｷﾞ)｣";
3040 IF PM%=34 THEN PRINT "｢ON(ﾋｮｳｼﾞ)/OFF(ｼｮｳｷｮ)｣";
3050 IF PM%=40 THEN PRINT "｢V(ﾚﾂﾕｳｾﾝ)/H(ｷﾞｮｳﾕｳｾﾝ)｣";
3060 IF PM%=42 THEN PRINT "｢";P$;"ﾚﾂﾉﾅｶﾞｻ (1-35)｣";
3070 IF PM%=44 THEN PRINT "｢V(ﾚﾂﾊﾞﾝｺﾞｳ)/H(ｷﾞｮｳﾊﾞﾝｺﾞｳ)｣";
3080 IF PM%=45 THEN PRINT "｢ｷﾞｮｳﾊﾞﾝｺﾞｳ (1-40)｣";
3090 IF PM%=46 THEN PRINT "｢ｷﾞｮｳ/ﾚﾂ/ｷﾞｮｳﾚﾂ ﾊﾞﾝｺﾞｳ｣";
3100 IF PM%=47 THEN PRINT "｢G(ｽﾍﾞﾃ)/D(ﾃﾞｰﾀ)/N(ｽｳﾁ)｣?";
3110 IF PM%=48 THEN PRINT "｢A(ank)/K(ｹｲｻﾝｼｷ)｣";
3120 IF PM%=50 THEN PRINT "｢ｸﾘｱｼﾃｲｲﾃﾞｽｶ?(Y/N)｣";
3130 IF PM%=60 THEN PRINT "｢ｺｳﾓｸﾊﾞﾝｺﾞｳ(ﾋﾀﾞﾘｼﾞｮｳﾀﾞﾝ｣";
3140 IF PM%=61 THEN PRINT "｢ｺｳﾓｸﾊﾞﾝｺﾞｳ(ﾐｷﾞｹﾞﾀﾞﾝ)｣";
3150 IF PM%=70 THEN PRINT "     ｢ﾌｧｲﾙ ﾒｲ｣";
3160 IF PM%=75 THEN PRINT "｢S(ｾｰﾌﾞ)/L(ﾛｰﾄﾞ)｣";
4999 RETURN
10000 '*** DATA INPUT SUB ***
10005 E%=3:DX%=3:DY%=GY%:CX%=LX%:CY%=LY%
10010 X%=DX%:Y%=DY%
10020 L%=TN%(CX%):GOSUB 1000
10030 IF P%=3 THEN GOSUB 11540:RETURN
10040 IF P$="" THEN 11000
```

```
10050 IF P%=2 THEN 10700
10060 IF TQ%(CX%,CY%)<>9 THEN IF P%=1 THEN TQ%(CX%,CY%)=1 ELSE TQ%(CX%,CY%)=0
10070 '
10080 D$(CX%,CY%)=P$
10090 GOTO 11000
10700 '
10710 IF P$=CHR$(28) THEN CF$="R":GOTO 10735
10720 IF P$=CHR$(29) THEN CF$="L":GOTO 10735
10730 IF P$=CHR$(30) THEN CF$="U" ELSE CF$="D"
10735 LOCATE 1,0:PRINT CF$;
10740 P$=""
11000 '
11010 GOSUB 11540
11020 IF CF$="U" THEN IF CY%=1 THEN 10010 ELSE DY%=DY%-1:CY%=CY%-1:IF CY%<LY% TH
EN LY%=LY%-1:GOSUB 1700:DY%=GY%:GOTO 10010 ELSE 10010
11030 IF CF$="D" THEN IF CY%=40 THEN 10010 ELSE DY%=DY%+1:CY%=CY%+1:IF CY%>RY% T
HEN LY%=LY%+1:GOSUB 1700:DY%=6:GOTO 10010 ELSE 10010
11040 IF CF$="R" THEN GOSUB 11300:IF QS$="E" THEN 10010 ELSE IF QS$="" THEN 1001
0 ELSE GOSUB 1500:GOSUB 1700:GOTO 10010
11050 GOSUB 11400:IF QS$="E" THEN 10010
11055 IF QS$="C" THEN GOSUB 1500:GOSUB 1700:GOTO 10010 ELSE 10010
11300 '
11310 QS$="":IF CX%=20 THEN QS$="E":RETURN
11320 DX%=DX%+TN%(CX%):CX%=CX%+1
11330 IF CX%<=RX% THEN RETURN
11340 QS$="C":DX%=DX%-TN%(LX%):LX%=LX%+1:GOSUB 1400
11350 IF CX%<=RX% THEN RETURN ELSE 11340
11400 '
11410 QS$="":IF CX%=1 THEN QS$="E":RETURN
11420 CX%=CX%-1:DX%=DX%-TN%(CX%)
11430 IF CX%>=LX% THEN RETURN
11440 LX%=LX%-1:GOSUB 1400:DX%=3:QS$="C":RETURN
11540 LOCATE DX%,DY%:DD$=D$(CX%,CY%):DL%=LEN(DD$)
11550 IF DL%>=TN%(CX%) THEN PRINT LEFT$(DD$,TN%(CX%)); ELSE IF TZ%(CX%)=1 THEN P
RINT DD$;STRING$(TN%(CX%)-DL%," "); ELSE PRINT STRING$(TN%(CX%)-DL%," ");DD$;
11560 RETURN
12000 '
12010 ZT%=1:GOSUB 1500:GOSUB 1700
12020 X%=3:Y%=0:L%=37:E%=1:GOSUB 1000
12030 IF P$="" THEN RETURN
12040 TI$=P$+STRING$(37-LEN(P$)," ")
12050 LOCATE 3,0:PRINT TI$;
12060 RETURN
14000 '
14010 ZQ%=1:LX%=1:LY%=1:CX%=1:CY%=1:GOSUB 1500:GOSUB 1700
14020 X%=3:Y%=GY%-1:II%=1:E%=3
14030 L%=TN%(II%):GOSUB 1000
14040 IF P%=3 THEN 14140
14050 IF P%=2 THEN 14150
14060 IF P$="" THEN 14090
14070 TN$(II%)=P$
14090 GOSUB 14220
14100 II%=II%+1:IF II%=21 THEN 14030
14110 IF II%<=RX% THEN X%=X%+TN%(II%-1):GOTO 14030 ELSE X%=X%+TN%(II%-1)-TN%(LX%
)
14120 LX%=LX%+1:GOSUB 1500:GOSUB 1700
14130 GOTO 14030
14140 GOSUB 14220:RETURN
14150 '
14160 IF P$=CHR$(30) OR P$=CHR$(31) THEN 14030
14170 IF P$=CHR$(28) THEN 14090
14180 GOSUB 14220
14190 IF II%=1 THEN 14030 ELSE II%=II%-1
14200 IF II%=>LX% THEN X%=X%-TN%(II%):GOTO 14030
14210 LX%=LX%-1:GOSUB 1500:GOSUB 1700:GOTO 14030
14220 LOCATE X%,Y%:DL%=LEN(TN$(II%)):IF DL%>TN%(II%) THEN PRINT LEFT$(TN$(II%),T
N%(II%)); ELSE IF TZ%(II%)=1 THEN PRINT TN$(II%);STRING$(TN%(II%)-DL%," "); ELSE
 PRINT STRING$(TN%(II%)-DL%," ");TN$(II%);
14230 RETURN
16000 '*** KEISANSIKI INPUT SUB ***
16005 E%=3:CX%=LX%:CY%=LY%:DT%=ZT%:ZT%=1:TB%=1:GOSUB 1500:GOSUB 1700:DX%=3:DY%=G
Y%
16010 X%=4:Y%=7:MX%=DX%:MY%=DY%:LOCATE 4,0:PRINT STRING$(25," ");:LOCATE 4,0:PRI
NT DK$(CX%,CY%);:LOCATE 4,7:PRINT STRING$(35," ");:L%=34
16020 GG%=1:GOSUB 1000:GG%=0:LOCATE 0,7:PRINT " ";
16030 IF P%=3 THEN 17500
16040 IF P$="" THEN 17000
```

```
16050 IF P%=2 THEN 16700
16060 GOSUB 17600:IF P$="//" THEN 17000 ELSE IF ER%=1 THEN LOCATE 0,7:PRINT "E";
:FOR BI%=1 TO 3:BEEP:NEXT BI%:GOTO 16010
16070 TQ%(CX%,CY%)=9
16080 DK$(CX%,CY%)=P$:D$(CX%,CY%)=""
16090 GOTO 17000
16700 '
16710 IF P$=CHR$(28) THEN CF$="R":GOTO 16735
16720 IF P$=CHR$(29) THEN CF$="L":GOTO 16735
16730 IF P$=CHR$(30) THEN CF$="U" ELSE CF$="D"
16735 LOCATE 1,0:PRINT CF$;
16740 P$="":GOTO 17000
17000 '
17010 GOSUB 17540
17020 IF CF$="U" THEN IF CY%=1 THEN 16010 ELSE DY%=DY%-1:CY%=CY%-1:IF CY%<LY% TH
EN LY%=LY%-1:GOSUB 1700:DY%=GY%:GOTO 16010 ELSE 16010
17030 IF CF$="D" THEN IF CY%=40 THEN 16010 ELSE DY%=DY%+1:CY%=CY%+1:IF CY%>RY% T
HEN LY%=LY%+1:GOSUB 1700:DY%=6:GOTO 16010 ELSE GOTO 16010
17040 IF CF$="R" THEN GOSUB 11300:IF QS$="E" THEN 16010 ELSE IF QS$="" THEN 1601
0 ELSE GOSUB 1500:GOSUB 1700:GOTO 16010
17050 GOSUB 11400:IF QS$="E" THEN 16010
17055 IF QS$="C" THEN GOSUB 1500:GOSUB 1700:GOTO 16010 ELSE 16010
17300 '
17500 '
17510 ZT%=DT%:TB%=0:GOSUB 1500:GOSUB 1700:GG%=0
17520 RETURN
17540 DD$=DK$(CX%,CY%):DL%=LEN(DD$):LOCATE DX%,DY%
17550 IF TQ%(CX%,CY%)=9 THEN IF DL%>=TN%(CX%) THEN PRINT LEFT$(DD$,TN%(CX%));:RE
TURN ELSE IF TZ%(CX%)=1 THEN PRINT DD$;STRING$(TN%(CX%)-DL%," ");:RETURN ELSE PR
INT STRING$(TN%(CX%)-DL%," ");DD$;:RETURN
17560 DD$=D$(CX%,CY%):DL%=LEN(DD$):IF DL%>=TN%(CX%) THEN PRINT LEFT$(DD$,TN%(CX%
));:RETURN ELSE IF TZ%(CX%)=1 THEN PRINT DD$;STRING$(TN%(CX%)-DL%," ");:RETURN E
LSE PRINT STRING$(TN%(CX%)-DL%," ");DD$;
17570 RETURN
17600 '
17610 ER%=0:DL%=LEN(P$):DI%=1
17615 IF LEFT$(P$,2)="//" THEN TQ%(CX%,CY%)=0:DK$(CX%,CY%)="":D$(CX%,CY%)="":RET
URN
17620 DC$=MID$(P$,DI%,1)
17630 IF DC$="@" THEN 17800
17640 IF DC$="&" THEN GOSUB 17990:IF ER%=1 THEN RETURN ELSE IF DC$="S" THEN 1770
0 ELSE IF DC$="D" OR DC$="I" THEN 17750 ELSE ER%=1:RETURN
17662 '
17665 IF DC$="+" OR DC$="-" THEN GOSUB 17990:IF ER%=1 THEN RETURN
17670 IF DC$>"/" AND DC$<":" THEN 17900
17673 IF DC$>"@" AND DC$<"U" THEN GOSUB 17990:IF ER%=1 THEN RETURN ELSE 17676 EL
SE ER%=1:RETURN
17676 IF DC$>"0" AND DC$<":" THEN DF$=DC$:GOSUB 17990:IF ER%=1 THEN ER%=0:RETURN
 ELSE 17680 ELSE ER%=1:RETURN
17680 IF DC$>"/" AND DC$<":" THEN DF$=DF$+DC$:IF DF$>"40" THEN ER%=1:RETURN ELSE
 GOSUB 17990:IF ER%=1 THEN ER%=0:RETURN
17683 IF DC$="+" OR DC$="-" OR DC$="/" OR DC$="*" THEN GOSUB 17990:IF ER%=1 THEN
 RETURN ELSE GOTO 17665 ELSE ER%=1:RETURN
17700 '
17703 GOSUB 17990:IF ER%=1 THEN RETURN
17706 IF DC$<"0" THEN ER%=1:RETURN
17710 IF DC$>"4" THEN ER%=1:RETURN
17713 GOSUB 17990:IF ER%=1 THEN RETURN
17716 IF DC$<>":" THEN ER%=1:RETURN
17720 GOSUB 17990:IF ER%=1 THEN RETURN
17730 GOTO 17630
17750 '
17753 GOSUB 17990:IF ER%=1 THEN RETURN
17756 IF DC$<>":" THEN ER%=1:RETURN
17760 GOSUB 17990:IF ER%=1 THEN RETURN
17770 GOTO 17630
17800 '
17802 HX$="":HY$="":NX$="":NY$="":DI%=DI%+1
17803 IF MID$(P$,DI%,2)="TL" THEN 17810
17806 IF MID$(P$,DI%,2)<>"AV" THEN ER%=1:RETURN
17810 DI%=DI%+1
17820 GOSUB 17990:IF ER%=1 THEN RETURN
17830 IF DC$<>"(" THEN ER%=1:RETURN
17833 GOSUB 17990:IF ER%=1 THEN RETURN
17836 IF DC$>"@" AND DC$<"U" THEN HX$=DC$:GOTO 17840 ELSE ER%=1:RETURN
17840 GOSUB 17990:IF ER%=1 THEN RETURN
```

```
17843 IF DC$>"0" AND DC$<":" THEN NX$=DC$:GOTO 17850 ELSE ER%=1:RETURN
17850 GOSUB 17990:IF ER%=1 THEN RETURN
17853 IF DC$>"/" AND DC$<":" THEN NX$=NX$+DC$:IF NX$>"40" THEN ER%=1:RETURN ELSE
 GOSUB 17990:IF ER%=1 THEN RETURN
17856 IF DC$<>":" THEN ER%=1:RETURN
17860 GOSUB 17990:IF ER%=1 THEN RETURN
17863 IF DC$<"A" THEN ER%=1:RETURN
17866 IF DC$>"T" THEN ER%=1:RETURN
17870 HY$=DC$:GOSUB 17990:IF ER%=1 THEN RETURN
17873 IF DC$<"1" THEN ER%=1:RETURN
17876 IF DC$>"9" THEN ER%=1:RETURN
17880 NY$=DC$:GOSUB 17990:IF ER%=1 THEN RETURN
17883 IF DC$>"/" AND DC$<":" THEN NY$=NY$+DC$:IF NY$>"40" THEN ER%=1:RETURN ELSE
 GOSUB 17990:IF ER%=1 THEN RETURN
17886 IF HX$<>HY$ THEN IF NX$<>NY$ THEN ER%=1:RETURN ELSE IF HX$>HY$ THEN ER%=1:
RETURN ELSE 17893 ELSE IF NX$>NY$ THEN ER%=1:RETURN
17893 IF DC$<>")" THEN ER%=1:RETURN
17896 GOSUB 17990:IF ER%=1 THEN ER%=0:RETURN ELSE ER%=1 RETURN
17900 '
17903 EB%=0
17910 GOSUB 17990:IF ER%=1 THEN ER%=0:RETURN
17912 IF DC$="." THEN IF EB%=0 THEN EB%=1:GOTO 17910 ELSE ER%=1:RETURN
17920 IF DC$>"/" AND DC$<":" THEN 17910 ELSE 17683
17990 '
17992 ER%=0:IF DI%>=DL% THEN ER%=1:RETURN
17994 DI%=DI%+1:DC$=MID$(P$,DI%,1)
17996 RETURN
18000 '***** CALC RTN *****
18010 PM%=40:GOSUB 2000:X%=4:Y%=7:L%=2:E%=1:GOSUB 1000
18020 IF P%=3 THEN RETURN
18030 IF P$="" THEN 18010
18040 IF P$="V" THEN GOSUB 18100 ELSE IF P$="H" THEN GOSUB 18200 ELSE 18010
18050 GOSUB 1700
18060 RETURN
18100 '
18110 FOR FI%=1 TO 20:FOR FJ%=1 TO 40:IF TQ%(FI%,FJ%)=9 THEN P$=DK$(FI%,FJ%):PL%
=LEN(P$):GOSUB 18500:NEXT FJ%:NEXT FI%:RETURN ELSE NEXT FJ%:NEXT FI%:RETURN
18200 '
18210 FOR FJ%=1 TO 40:FOR FI%=1 TO 20:IF TQ%(FI%,FJ%)=9 THEN P$=DK$(FI%,FJ%):PL%
=LEN(P$):GOSUB 18500:NEXT FI%:NEXT FJ%:RETURN ELSE NEXT FI%:NEXT FJ%:RETURN
18500 '
18510 MI%=0:MS%=0:MD%=0:AN#=0:AW#=0:AC%=0:CC%=0
18520 GOSUB 19000
18530 IF DC$="&" THEN GOSUB 19000:IF DC$="I" THEN MI%=1:AC%=AC%+1:GOTO 18520 ELS
E IF DC$="D" THEN MD%=1:AC%=AC%+1:GOTO 18520 ELSE GOSUB 19000:MS%=VAL(DC$):AC%=A
C%+1:GOTO 18520
18540 IF DC$="@" THEN 18900
18600 '
18610 GOSUB 18750:AN#=AW#
18620 IF PE%=1 THEN 18800
18630 DF$=DC$:GOSUB 19000
18640 GOSUB 18750
18670 '
18690 IF DF$="+" THEN AN#=AN#+AW#
18700 IF DF$="-" THEN AN#=AN#-AW#
18710 IF DF$="*" THEN AN#=AN#*AW#
18720 IF DF$="/" THEN IF AW#<>0 THEN AN#=AN#/AW#
18730 IF PE%=1 THEN 18800 ELSE 18630
18750 '
18755 WC$="":PE%=0:IF DC$>"@" AND DC$<"U" THEN GOSUB 19200:AW#=VAL(D$(AI%,AJ%)):
RETURN
18760 IF DC$="+" OR DC$="-" THEN WC$=DC$:AC%=AC%+1
18765 WC$=WC$+MID$(P$,AC%,1):AC%=AC%+1:IF AC%>PL% THEN PE%=1:GOTO 18780
18770 IF MID$(P$,AC%,1)>"/" THEN 18765
18775 IF MID$(P$,AC%,1)="." THEN 18765
18780 AW#=VAL(WC$):DC$=MID$(P$,AC%,1):RETURN
18800 '
18810 KC$=STR$(AN#)
18820 GOSUB 18990
18830 IF MI%=1 THEN IF KI%<>0 THEN IF MD%=1 THEN AN#=AN#+.5:KC$=STR$(AN#):GOSUB
18990:IF KI%<>0 THEN KC$=LEFT$(KC$,KI%-1):GOTO 18860 ELSE 18860 ELSE KC$=LEFT$(K
C$,KI%-1):GOTO 18860 ELSE 18860
18840 IF MS%<>0 THEN IF LEN(KC$)-KI%<=MS% THEN 18860 ELSE IF MD%=1 THEN AN#=AN#+
(.1^(MS%+1)*5):KC$=STR$(AN#):GOSUB 18990:IF LEN(KC$)-KI%<=MS% THEN 18860 ELSE KC
$=LEFT$(KC$,KI%+MS%):GOTO 18860 ELSE KC$=LEFT$(KC$,KI%+MS%):GOTO 18860
18850 '
```

```
18860 D$(FI%,FJ%)=KC$
18870 RETURN
18900 '
18910 GOSUB 19000:AC%=AC%+3:GOSUB 19500:IF DC$="A" THEN 18950
18920 IF AI%=BI% THEN FOR SJ%=AJ% TO BJ%:IF TQ%(AI%,SJ%)<>0 THEN AN#=AN#+VAL(D$(
AI%,SJ%)):NEXT ELSE NEXT ELSE FOR SI%=AI% TO BI%:IF TQ%(SI%,AJ%)<>0 THEN AN#=AN#
+VAL(D$(SI%,AJ%)):NEXT ELSE NEXT
18930 GOTO 18800
18950 IF AI%=BI% THEN FOR SJ%=AJ% TO BJ%:IF TQ%(AI%,SJ%)<>0 THEN AN#=AN#+VAL(D$(
AI%,SJ%)):CC%=CC%+1:NEXT ELSE NEXT ELSE FOR SI%=AI% TO BI%:IF TQ%(SI%,AJ%)<>0 TH
EN AN#=AN#+VAL(D$(SI%,AJ%)):CC%=CC%+1:NEXT ELSE NEXT
18960 IF CC%=0 THEN 18800
18970 AN#=AN#/CC%:GOTO 18800
18990 FOR KI%=LEN(KC$) TO 1 STEP -1:IF MID$(KC$,KI%,1)="." THEN RETURN ELSE NEXT
:KI%=0:RETURN
19000 '
19010 PE%=0:IF AC%>PL% THEN PE%=1:RETURN
19020 AC%=AC%+1:DC$=MID$(P$,AC%,1)
19030 RETURN
19200 '
19210 AI%=ASC(DC$)-64:WC$="":PE%=0
19220 AC%=AC%+1:IF AC%>PL% THEN PE%=1:AJ%=VAL(WC$):RETURN
19225 DC$=MID$(P$,AC%,1)
19230 IF DC$<"0" THEN AJ%=VAL(WC$):RETURN
19240 IF DC$>"9" THEN AJ%=VAL(WC$):RETURN ELSE WC$=WC$+DC$:GOTO 19220
19500 '
19510 AI%=ASC(MID$(P$,AC%,1))-64:AC%=AC%+1:WC$=MID$(P$,AC%,1):AC%=AC%+1
19520 IF MID$(P$,AC%,1)<>":" THEN WC$=WC$+MID$(P$,AC%,1):AC%=AC%+1
19530 AJ%=VAL(WC$):AC%=AC%+1
19540 BI%=ASC(MID$(P$,AC%,1))-64:AC%=AC%+1:WC$=MID$(P$,AC%,1):AC%=AC%+1
19550 IF MID$(P$,AC%,1)<>")" THEN WC$=WC$+MID$(P$,AC%,1)
19560 BJ%=VAL(WC$):RETURN
20000 '
20010 PM%=32:GOSUB 2000:X%=4:Y%=7:L%=2:E%=1:GOSUB 1000
20012 IF P%=3 THEN RETURN
20014 IF P%=1 OR P%=2 OR P$="" THEN 20010
20020 IF P$<"A" THEN 20010
20030 IF P$>"U" THEN 20010
20040 PU%=ASC(P$)-64
20050 PM%=42:GOSUB 2000:X%=3:Y%=7:L%=3:GOSUB 1000
20060 IF P%=0 OR P%=2 OR P$="" THEN 20050
20070 IF P%=3 THEN 20010
20080 IF LEN(P$)=1 THEN IF P$<"1" THEN 20050 ELSE PM%=VAL(P$) ELSE PM%=VAL(P$):I
F PM%=0 THEN 20050 ELSE IF PM%>35 THEN 20050
20090 TN%(PU%)=PM%:GOSUB 1500:GOSUB 1700:GOTO 20010
22000 '
22010 PM%=44:GOSUB 2000:X%=4:Y%=7:L%=2:E%=1:GOSUB 1000
22015 IF P%=3 THEN RETURN
22020 IF P%=1 OR P%=2 OR P$="" THEN 22010
22040 IF P$="H" THEN 22500
22050 IF P$<>"V" THEN 22010
22060 PM%=32:GOSUB 2000:X%=4:Y%=7:L%=2:GOSUB 1000
22070 IF P%=3 THEN 22010
22080 IF P%=1 OR P%=2 OR P$="" THEN 22060
22090 IF P$>"@" AND P$<"U" THEN PD%=ASC(P$)-64:TN%(PD%)=1 ELSE 22060
22100 FOR I%=1 TO 40:D$(PD%,I%)="I":TQ%(PD%,I%)=0:NEXT I%
22110 IF PD%<LX% THEN 22010
22120 IF PD%>RX% THEN 22010
22130 GOSUB 1500:GOSUB 1700:GOTO 22010
22500 '
22510 PM%=45:GOSUB 2000:X%=4:Y%=7:L%=3:GOSUB 1000
22520 IF P%=3 THEN 22010 ELSE IF P%=0 OR P%=2 THEN 22510
22530 PC%=VAL(P$)
22535 IF PC%<1 THEN 22510 ELSE IF PC%>40 THEN 22510
22540 FOR I%=1 TO 20:D$(I%,PC%)=STRING$(TN%(I%),"-"):TQ%(I%,PC%)=0:NEXT I%
22550 IF PC%<LY% THEN 22010
22560 IF PC%>RY% THEN 22010
22570 GOTO 22130
24000 '
24010 PM%=30:GOSUB 2000
24020 X%=4:Y%=7:L%=3:E%=1:GOSUB 1000
24025 IF P%=3 THEN RETURN
24030 IF P$="" THEN IF PM%=31 THEN 24010 ELSE PM%=31:GOSUB 2000:GOTO 24020
24040 IF P$="ZT" THEN PM%=34:GOSUB 2000:GOSUB 1000:IF P%=3 THEN 24010 ELSE IF P$
="ON" THEN ZT%=1:GOSUB 1500:GOSUB 1700:RETURN ELSE IF P$="OFF" THEN ZT%=0:GOSUB
1500:GOSUB 1700:RETURN ELSE P$="ZT":GOTO 24040
```

```
24050 IF P$="ZN" THEN PM%=34:GOSUB 2000:GOSUB 1000:IF P%=3 THEN 24010 ELSE IF P$
="ON" THEN ZQ%=1:GOSUB 1500:GOSUB 1700:RETURN ELSE IF P$="OFF" THEN ZQ%=0:GOSUB
1500:GOSUB 1700:RETURN ELSE P$="ZN":GOTO 24050
24060 IF P$="ZA" THEN PM%=34:GOSUB 2000:GOSUB 1000:IF P%=3 THEN 24010 ELSE IF P$
="ON" THEN ZA%=1:GOSUB 1500:GOSUB 1700:RETURN ELSE IF P$="OFF" THEN ZA%=0:GOSUB
1500:GOSUB 1700:RETURN ELSE P$="ZA":GOTO 24060
24070 IF P$<>"ZF" THEN 24010
24080 PM%=32:GOSUB 2000:GOSUB 1000
24090 IF P%=3 THEN 24010
24100 IF P$>"@" AND P$<"U" THEN WP%=ASC(P$)-64 ELSE 24080
24120 PM%=33:GOSUB 2000:GOSUB 1000:IF P%=3 THEN 24080 ELSE IF P$="R" THEN TZ%(WP
%)=0:GOSUB 1500:GOSUB 1700:GOTO 24080 ELSE IF P$="L" THEN TZ%(WP%)=1:GOSUB 1500:
GOSUB 1700:GOTO 24080 ELSE 24120
26000 '
26002 PM%=75:GOSUB 2000:X%=4:Y%=7:L%=2:E%=1:GOSUB 1000:IF P%=3 THEN RETURN
26004 IF P$="S" THEN 26010 ELSE IF P$="L" THEN 26500 ELSE 26002
26010 PM%=70:GOSUB 2000:L%=7:GOSUB 1000:IF P%=3 THEN 26002
26020 IF P%<>0 OR P$="" THEN 26010
26030 IF LEN(P$)>6 THEN 26010
26040 FF$=P$
26050 PM%=60:GOSUB 2000:GOSUB 1000:IF P%=3 THEN 26010
26060 GOSUB 42000:IF RE%=1 THEN 26050
26070 V1%=Q1%:V2%=Q2%
26080 PM%=61:GOSUB 2000:GOSUB 1000:IF P%=3 THEN 26010
26090 GOSUB 42000:IF RE%=1 THEN 26050
26100 V3%=Q1%:V4%=Q2%
26110 OPEN "CAS:"+FF$ FOR OUTPUT AS#1:PRINT#1,V1%;V2%;V3%;V4%;",";TI$;",";
26120 FOR ZI%=V1% TO V3%:PRINT#1,TN$(ZI%);",";TN%(ZI%);TZ%(ZI%);",";:NEXT ZI%
26130 FOR ZJ%=V2% TO V4%:FOR ZI%=V1% TO V3%
26140 PRINT#1,D$(ZI%,ZJ%);",";DK$(ZI%,ZJ%);",";TQ%(ZI%,ZJ%);",";
26150 NEXT ZI%,ZJ%:CLOSE#1:RETURN
26500 PM%=70:GOSUB 2000:L%=7:GOSUB 1000:IF P%=3 THEN 26002
26510 IF P%<>0 OR P$="" THEN 26500
26520 IF LEN(P$)>6 THEN 26500
26530 OPEN "CAS:"+P$ FOR INPUT AS#1
26535 LOCATE 11,7:PRINT P$;" ﾛｰﾄﾞﾁｭｳ";
26540 INPUT#1,V1%,V2%,V3%,V4%,TI$
26550 FOR ZI%=V1% TO V3%:INPUT#1,TN$(ZI%),TN%(ZI%),TZ%(ZI%):NEXT ZI%
26560 FOR ZJ%=V2% TO V4%:FOR ZI%=V1% TO V3%
26570 INPUT#1,D$(ZI%,ZJ%),DK$(ZI%,ZJ%),TQ%(ZI%,ZJ%)
26580 NEXT ZI%,ZJ%:CLOSE#1:GOSUB 1500:GOSUB 1700:RETURN
28000 '
28010 PM%=60:GOSUB 2000:X%=4:Y%=7:L%=3:E%=1:GOSUB 1000:IF P%=3 THEN RETURN
28020 GOSUB 42000:IF RE%=1 THEN 28010 ELSE V1%=Q1%:V2%=Q2%
28030 PM%=61:GOSUB 2000:GOSUB 1000:IF P%=3 THEN 28010
28040 GOSUB 42000:IF RE%=1 THEN 28030 ELSE V3%=Q1%:V4%=Q2%:PB%=0:LV%=0
28055 FOR LO%=V1% TO V3%:LV%=LV%+TN%(LO%):NEXT
28057 IF LV%+3>79 THEN CB$="Q" ELSE CB$="N"
28060 LPRINT CHR$(27);"N";"   ";TI$;CHR$(27);CB$:LPRINT:LPRINT "   ";
28062 FOR OL%=V1% TO V3%
28063 IF TZ%(OL%)=1 THEN IF LEN(TN$(OL%))>=TN%(OL%) THEN LPRINT LEFT$(TN$(OL%),T
N%(OL%)); ELSE LPRINT TN$(OL%);STRING$(TN%(OL%)-LEN(TN$(OL%))," ");:GOTO 28068
28066 IF LEN(TN$(OL%))>=TN%(OL%) THEN LPRINT RIGHT$(TN$(OL%),TN%(OL%)); ELSE LPR
INT STRING$(TN%(OL%)-LEN(TN$(OL%))," ");TN$(OL%);
28068 NEXT OL%:LPRINT
28070 FOR OV%=V2% TO V4%:LPRINT RIGHT$(STR$(OV%),2);" ";
28080 FOR OL%=V1% TO V3%
28090 IF TZ%(OL%)=1 THEN IF LEN(D$(OL%,OV%))>=TN%(OL%) THEN LPRINT LEFT$(D$(OL%,
OV%),TN%(OL%));:GOTO 28110 ELSE LPRINT D$(OL%,OV%);STRING$(TN%(OL%)-LEN(D$(OL%,O
V%))," ");:GOTO 28110
28100 IF LEN(D$(OL%,OV%))>=TN%(OL%) THEN LPRINT RIGHT$(D$(OL%,OV%),TN%(OL%)); EL
SE LPRINT STRING$(TN%(OL%)-LEN(D$(OL%,OV%))," ");D$(OL%,OV%);
28110 NEXT OL%:LPRINT:NEXT OV%:RETURN
30000 '
30010 PM%=46:GOSUB 2000:X%=4:Y%=7:L%=3:E%=3:GOSUB 1000
30020 IF P%=3 THEN RETURN
30030 IF P%=2 OR P$="" THEN 30010
30040 IF P%=1 THEN 30500
30050 IF LEN(P$)=1 THEN 30300
30060 '
30070 BC$=LEFT$(P$,1):BD$=RIGHT$(P$,LEN(P$)-1)
30080 IF BC$<"A" THEN 30010 ELSE IF BC$>"T" THEN 30010
30100 IF LEN(BD$)=1 THEN IF BD$<"0" THEN 30010 ELSE IF BD$>"9" THEN 30010 ELSE 3
0105 ELSE IF LEFT$(BD$,1)<"0" THEN 30010 ELSE IF LEFT$(BD$,1)>"5" THEN 30010 ELS
E IF RIGHT$(BD$,1)<"0" THEN 30010 ELSE IF RIGHT$(BD$,1)>"9" THEN 30010
30103 IF VAL(BD$)>35 THEN 30010
```

```
30105 LX%=ASC(BC$)-64
30110 LY%=VAL(BD$):GOSUB 1500:GOSUB 1700:GOTO 30010
30300 '
30310 IF P$<"A" THEN 30010 ELSE IF P$>"T" THEN 30010
30320 LX%=ASC(P$)-64:GOSUB 1500:GOSUB 1700:GOTO 30010
30500 '
30510 PC%=VAL(P$):IF PC%<1 THEN 30010 ELSE IF PC%>35 THEN 30010
30520 LY%=PC%:GOSUB 1700:GOTO 30010
34000 '
34010 PM%=47
34020 GOSUB 2000:X%=4:Y%=7:L%=2:E%=1:GOSUB 1000
34030 IF P%=3 THEN RETURN
34040 IF P%=1 OR P%=2 THEN 34020
34050 IF P$="" THEN IF PM%=47 THEN PM%=48:GOTO 34020 ELSE PM%=47:GOTO 34020
34060 IF P$="G" THEN 34100 ELSE IF P$="D" THEN 34200 ELSE IF P$="N" THEN 34300 E
LSE IF P$="A" THEN 34400 ELSE IF P$="K" THEN 34500 ELSE 34020
34100 '
34110 PM%=50:GOSUB 2000:X%=4:Y%=7:L%=2:E%=1:GOSUB 1000
34120 IF P%=3 THEN RETURN
34130 IF P$<>"Y" THEN RETURN
34140 RUN
34200 '
34210 FOR BJ%=1 TO 40:FOR BI%=1 TO 20
34220 D$(BI%,BJ%)="":IF TQ%(BI%,BS%)<>9 THEN TQ%(BI%,BJ%)=0
34230 NEXT BI%,BJ%:GOSUB 1700:GOTO 34010
34300 '
34310 FOR BJ%=1 TO 40:FOR BI%=1 TO 20
34320 IF TQ%(BI%,BJ%)=1 THEN D$(BI%,BJ%)="":TQ%(BI%,BJ%)=0
34330 NEXT BI%,BJ%:GOSUB 1700:GOTO 34010
34400 '
34410 FOR BJ%=1 TO 40:FOR BI%=1 TO 20
34420 IF TQ%(BI%,BJ%)=0 THEN D$(BI%,BJ%)=""
34430 NEXT BI%,BJ%:GOSUB 1700:GOTO 34010
34500 '
34510 FOR BJ%=1 TO 40:FOR BI%=1 TO 20
34520 IF TQ%(BI%,BJ%)=9 THEN TQ%(BI%,BJ%)=0:DK$(BI%,BJ%)="":D$(BI%,BJ%)=""
34530 NEXT BI%,BJ%:GOSUB 1700:GOTO 34010
42000 '
42010 RE%=0:IF P$="" OR LEN(P$)=1 OR LEN(P$)>3 THEN 42100
42020 IF LEFT$(P$,1)<"A" THEN 42100
42030 IF LEFT$(P$,1)>"T" THEN 42100 ELSE Q1%=ASC(LEFT$(P$,1))-64
42040 IF LEN(P$)=2 THEN IF RIGHT$(P$,1)<"1" OR RIGHT$(P$,1)>"9" THEN 42100 ELSE
Q2%=VAL(RIGHT$(P$,1)):RETURN
42050 SA$=RIGHT$(P$,2)
42060 IF LEFT$(SA$,1)<"0" OR LEFT$(SA$,1)>"5" THEN 42100
42070 IF RIGHT$(SA$,1)<"0" OR RIGHT$(SA$,1)>"9" THEN 42100
42080 Q2%=VAL(SA$):IF Q2%>40 THEN 42100 ELSE RETURN
42100 RE%=1:RETURN
59000 '*** INITIAL SUB ***
59003 CLS:LOCATE 0,0:PRINT "[ALPHA SOFT]          Copyright by NJK";
59005 LOCATE 10,3:PRINT "** ｼﾊﾞﾗｸ ｵﾏﾁｸﾀﾞｻｲ **";
59010 TI$=STRING$(37," ")
59020 FOR ZI%=1 TO 20:TN$(ZI%)="          ":TN%(ZI%)=7:TZ%(ZI%)=0
59040 NEXT ZI%
59050 TS%=0:TT%=0
59070 RETURN
```

エラーコード表

エラーメッセージ	コード	N-BASIC, N_{88}-Disk BASICのメッセージ（エラー名称）	意味
? AO Error	53	File already open (Already Open)	同じファイルを二度OPENした。
? AT Error (Disk)	63	Bad allocation table (なし)	ディスクのFATが壊れている。
? BM Error (Disk)	60	Bad file name (なし)	ファイル名が適当でない。
? BN Error	51	Bad file number (Bad file Number)	ファイル番号が適当でない。
? BO Error	23	Communications Buffer overflow (Buffer Overflow)	入力バッファが一杯になった。
? BS Error	9	Subscript out of range (Bad Subscript)	配列の添字が適当でない。
? CF Error	58	File not open (Closed File)	ファイルがまだOPENされていない。
? CN Error	17	Can't Continue (Continue Not possible)	CONTでプログラム実行再開ができない。
? DD Error	10	Duplicate Definition (Duplicate Definition)	同じ配列を二度宣言した。
? DF Error (Disk)	62	Disk full (なし)	ディスクが一杯で書き込めない。
? DN Error (Disk)	64	Bad drive number (なし)	ドライブ指定が適当でない。
? DS Error	56	Direct Statement in file (Direct Statement in file)	アスキー形式のファイルが読み込めない。
? DU Error	25	なし (Device Unavailable)	指定デバイスが使用できない。
? EF Error	54	Input past end (End of File)	ファイル中のデータを読み尽した。
? FC Error	5	Illegal function call (illegal Function Call)	命令や関数の使い方が適当でない。
? FE Error (Disk)	61	File already exists (なし)	NAMEで既存のファイル名を使った。
? FF Error	52	File not Found (File not Found)	指定したファイル名が見つからない。
? FL Error	57	なし (Filing Limit)	ファイルが多すぎる。
? FW Error (Disk)	67	File write protected (なし)	ファイルに書き込み禁止属性が付いている。
? ID Error	12	Illegal direct (Illegal Direct)	ダイレクトモードで使えない命令を使った。
? IE Error	50	Internal error (Internal Error)	BASIC内部にエラーが生じた。
? IO Error	24	I/O error (I/O error)	入出力中にエラーが生じた。

エラーメッセージ	コード	N-BASIC, N_{88}-Disk BASICのメッセージ（エラー名称）	意　味
? LS　Error	15	String too long (Long String)	文字型変数の内容が255文字を越えた。
? MO　Error	22	Missing operand (Missing Operand)	必要なパラメータが欠けている。
? NF　Error	1	NEXT without FOR (Next without For)	FOR文がないのにNEXT文がある。
? NM　Error	55	Bad file name (file Name Mismatch)	ファイル名が適当でない。
? NR　Error	19	No RESUME (No Resume)	エラー処理ルーチンにRESUME文がない。
? OD　Error	4	Out of data (Out of Data)	読むべきデータが足りない。
? OM　Error	7	Out of memory (Out of Memory)	メモリが足りない。
? OS　Error	14	Out of string space (Out of String space)	文字列格納用のメモリ領域が足りない。
? OV　Error	6	Overflow (OVerflow)	数値が大きすぎる。
? PC　Error	59	なし (PC-8001)	PC-8001の命令を使っている。
? RD　Error (Disk)	66	Rename across disks (なし)	異なるドライブ間でNAMEを実行した。
? RG　Error	3	RETURN without GOSUB (Return without Gosub)	GOSUB文がないのにRETURN文がある。
? RW　Error	20	RESUME without error (Resume Without error)	エラーがないのにRESUME文がある。
? SN　Error	2	Syntax error (SyNtax error)	文の記述が間違っている。
? ST　Error	16	String formula too complex (String formula Too complex)	文字式が複雑すぎる。
? TM　Error	13	Type mismatch (Type Mismatch)	変数や定数の型が合わない。
? TS　Error	65	Bad track／sector (なし)	トラック，セクター番号の指定が適当でない。
? UE　Error	21	Unprintable error (Unprintable Error)	メッセージの定義されていないエラー。
? UF　Error	18	Undefined user function (Undefined user Function)	未定義のユーザ関数を呼んだ。
? UL　Error	8	Undefined line number (Undefined Line number)	指定した行が定義してない。
? ／0　Error	11	Division by Zero (division by zero)	0による除算を行った。

索　引

日本電気ハンドヘルド・コンピュータ

PC-8201 活用の基礎

昭和58年12月20日　初版発行　定価2300円

著　作　日本情報研究センター
斉藤紀美子
発行人　黒川　巌
編集人　坪田良彦
発行所　株式会社　学習研究社
東京都大田区上池台4丁目40番5号
郵便番号　145　振替口座　東京⑧142930
電話　東京 03（720）1111（大代表）
印刷所　中央精版印刷株式会社・株式会社　金羊社

★この本の内容，製本に関するお問い合わせは，下記あてお願いします。
文書は，(〒145）東京都大田区上池台4丁目40番5号
学研　お客さま相談センター「PC-8201 活用の基礎」係
電話は，東京　03（720）1111（大代表）

166 707
ISBN 4-05-100711-1 C2055 ¥2300E

学研



学研

166 707
ISBN4-05-100711-1 C2055 ¥2300E 定価2,300円